MULTI-STAGE DISCRETE OPTIMIZATION UNDER UNCERTAINTY AND LOT-SIZING

By

ZHILI ZHOU

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2010

To my parents Guifeng Wang and Yaozong Zhou

for their love, support, and understanding

ACKNOWLEDGMENTS

I have worked with a great number of people whose contribution in assorted ways to the successful completion of this dissertation. It is a pleasure to convey my gratitude to them all in my humble acknowledgment.

I would like to record my gratitude to my supervisor, Dr. Yongpei Guan, for his supervision, advice, and guidance from the initial to the final stage of this research. He provided me unflinching encouragement and support in various ways. Dr. Guan taught me how to question thoughts and express ideas. His patience and support helped me overcome many crisis situations and finish this dissertation.

I gratefully acknowledge Dr. Cole Smith for his thought-provoking discussion and timely help during the writing of this dissertation. I wish to thank Dr. Joseph Hartman for the insightful advice on research and valuable mentor time. I am thankful to Dr. William Hager for broadening my view of mathematical programming. I am much indebted to Dr. Theodore Trafalis for his valuable advice in scientific discussions.

Many thanks go to my friends. They have helped me stay sane through these difficult years. Their support and care helped me to overcome setbacks and to stay focused on my graduate study. I greatly value their friendship and I deeply appreciate their belief in me. Especially, I am obliged to Natassia Brenkus, Belle Brenkus, and Tachun Lin.

Finally, I would like to express my heart-felt gratitude to my parents, Guifeng Wang and Yaozong Zhou. My family has been a constant source of love, concern, support, and strength all these years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

MULTI-STAGE DISCRETE OPTIMIZATION UNDER UNCERTAINTY
AND LOT-SIZING

By

Zhili Zhou

August 2010

Chair: Yongpei Guan
Major: Industrial and Systems Engineering

Multi-stage robust optimization and stochastic programming are two approaches for multi-stage decision making under data uncertainty. In this dissertation, three problems on multi-stage robust optimization and stochastic programming are discussed.

First, we consider a robust lot-sizing problem as an example to analyze multi-stage robust integer programming problems. In the robust lot-sizing problem setting, we consider the cases in which severe events may happen such that the normal process will be disrupted. Our objective is to provide a robust schedule such that the total cost is minimized under the worst case scenario. This problem can be formulated as a multi-stage robust integer programming problem. Several cases are studied and corresponding algorithms are developed. Our preliminary study verifies the effectiveness of our approaches.

Second, we consider two-stage stochastic uncapacitated lot-sizing problems with deterministic demands and Wagner-Whitin costs. We develop extended formulations in the higher dimensional space that can provide integral solutions by showing that their constraint matrices are totally unimodular. For the case without backlogging, we provide the convex hull description of the problem in the original space by projecting the extended formulation to the original space. For the case with backlogging, we provide a tighter extended formulation by projecting the extended formulation to a lower dimensional space.

Third, we study a general stochastic dynamic knapsack polytope. We apply the pairing, mixing, and lifting schemes to the stochastic dynamic knapsack set and obtain strong valid inequalities. We investigate the algorithmic and implementation issues for the effective and efficient generation of lifted valid inequalities of the stochastic dynamic knapsack polytope in a parallel computing environment. The speedup, communication overhead, load balance, and effectiveness in closing the integrality gap for stochastic dynamic knapsack polytope are studied. Computational experiments show the effectiveness of our proposed approaches.

CHAPTER 1
INTRODUCTION

Lot-sizing problems have been studied extensively during the last four decades. The deterministic single item lot-sizing problem is to decide a production plan for a product to satisfy demands for a fixed time horizon (i.e., T periods) while minimizing total costs.

The deterministic lot-sizing problem cannot provide robust production plans in the presence of uncertainty. As such, the deterministic decision making may yield unsatisfactory decisions. Therefore, we investigate stochastic programming and robust optimization and analyze their roles in decision making and their application in lot-sizing problems.

The goal of this dissertation is to analyze uncertainties in multi-stage discrete optimization programs. We investigate multi-stage robust optimization and adopt multi-stage stochastic programming approaches for mixed-integer programming problems through the study of lot-sizing problems.

Before we describe the outline of this dissertation, we introduce the following technique backgrounds to be used in the following chapters.

1.1 Stochastic Programming

Stochastic programming is an approach to process problems where the data incorporated into the objective or constraints are uncertain. Uncertainty is usually characterized by a probability distribution on the parameters. We seek a solution that is feasible and optimizes a given objective function. Two-stage stochastic programs are widely applied and studied stochastic programming models. The decision maker takes some actions on the first stage, after which a random event occurs affecting the outcome of the first stage decision.

1.1.1 Two-Stage Stochastic Programming

The study of stochastic programming was originated by Danzig (1955) and Beale (1955). They investigated the classical two-stage stochastic linear programming which

can be formulated as

$$\begin{aligned} \min\ z &= c^T x + E_\omega(\mathcal{Q}(x,\omega)) \\ s.t.\ Ax &= b, \\ x &\geq 0, \end{aligned} \tag{1–1}$$

where

$$\begin{aligned} \mathcal{Q}(x,\omega) = \min\ & f(\omega)^T y(\omega) \\ s.t.\ \ & T(\omega)x + W(\omega)y(\omega) = h(\omega), \\ & y(\omega) \geq 0. \end{aligned} \tag{1–2}$$

In this type of mathematical model, the data uncertainty can be represented as random variables ω. The particular values of random variables are known after the realization of random experiments. The decision variables are divided into two types: the first stage decision variable and the second stage decision variable. The first stage variable, denoted as x, has to be decised before the random experiment, and the period when this variable is taken is called the first stage. The second stage variable, denoted as y, can be determined after the realization of the random parameters. The corresponding period is called the second stage. The sequence of events and decisions is as

$$\text{Decision on } x \rightarrow \text{Observation on } \omega \rightarrow \text{Decision on } y(x,\omega).$$

Formulations (1–1) and (1–2) are considered as the first stage and the second stage problems, respectively. The matrix A and the vector b are deterministic parameters. The recourse function $\mathcal{Q}(x,\omega)$ defines the expected second stage value function. The matrix $T(\omega)$, vectors $h(\omega)$ and $f(\omega)$, and the transition matrix $W(\omega)$ can be random.

Several solution methods have been introduced for (1–1) and (1–2) with finite distribution. Among them, decomposition methods are important for solving large scale

two-stage stochastic programming problems. They can be categorized according to fundamental strategies as either outer linearization or inner linearization. Danzig and Mandasky (1961) applied Danzig-Wolfe decomposition (Dantzig and Wolfe 1960) to solve the dual of two-stage stochastic linear programming problem using inner linearization. Van Slyke and Wets (1969) developed the two-stage L-shaped method for solving the primal problem using outer linearization that is a form of Benders' decomposition (Benders 1962). Kall (1976) and Strazicky (1962) presented another dual method based on basis factorization. A method based on discrete distributions and splines was proposed by Wets 1974, 1983. For the continuous sample space, Birge and Louveaux (1997), Shapiro (2000), and Shapiro and de Mello (2001) proposed the Monte Carlo sampling method with finitely many scenarios. In general, Ahmed and Shapiro (2002) represented that the larger the number of scenarios, the more accurate is the provided model. This, in turn, makes the resulting formulation very large.

1.1.2 Multi-Stage Stochastic Programming

The multi-stage stochastic programming problem is an extension of the two-stage stochastic program in the multi-stage setting. Based on available information at each time period, multi-stage programs model problems where decisions should be made sequentially in certain time periods.

Birge and Louveaux (1997) and Louveaux and Schultz (2003) introduced the special stochastic scenario tree structure for the deterministic equivalent multi-stage stochastic linear programming. The structure can be interpreted as a scenario tree with T stages (or levels) where node i in stage t of the tree constitutes the state of the world that can be distinguished by information available up to time stage t. We use $\mathcal{T} = \mathcal{T}(0) = (\mathcal{V},\ \mathcal{E}) = (\mathcal{V}(0),\ \mathcal{E}(0))$ to represent the whole scenario tree. The set of leaf nodes in $\mathcal{V}$ is denoted as $\mathcal{L}$. Node i, $i \in \mathcal{V}$, $i \neq 0$ (the root node indexed as $i = 0$), has a unique parent $a(i)$. Node i, $i \in \mathcal{V} \setminus \mathcal{L}$ is the root node of a subtree $\mathcal{T}(i) = (\mathcal{V}(i),\ \mathcal{E}(i))$, which contains all descendants of node i, and has AN immediate children set $\mathcal{C}(i)$, i.e.,

$\mathcal{C}(i) = \{j : \ a(j) = i\}$. $\mathcal{L}(i)$ denotes the leaf nodes of the subtree $\mathcal{T}(i)$. The set of nodes on the path from the root node to node i is denoted by $\mathcal{P}(i)$. If $i \in \mathcal{L}$, $\mathcal{P}(i)$ corresponds to a scenario, and represents a joint realization of the problem parameters over this scenario. We define $\mathcal{P}(i,j) = \{k : k \in \mathcal{P}(j) \cap \mathcal{V}(i)\}$, thus $\mathcal{P}(i) = \mathcal{P}(0,i)$. $t(i)$ denotes the time stage(or level) of node i, i.e., $t(i) = |\mathcal{P}(i)|$. The probability associated with the state represented by node i is p_i. Let $\mathcal{V}(i)$ be the subtree with node i as root node. Figure 1-1 shows a multi-stage stochastic scenario tree model.

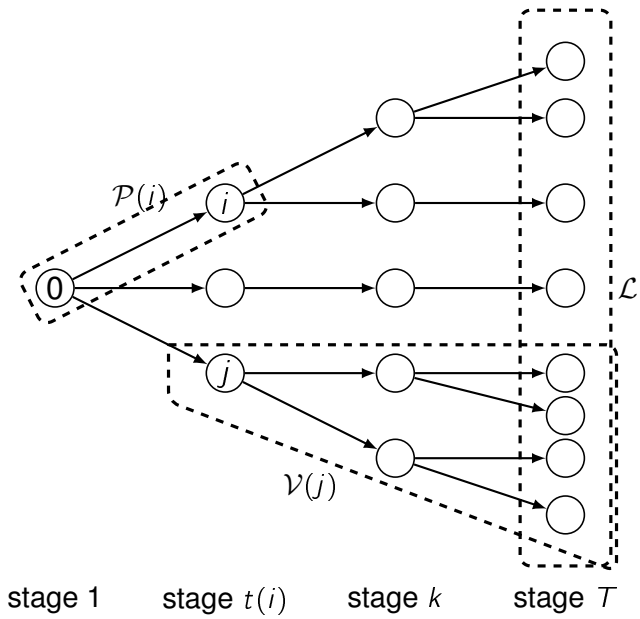


Figure 1-1. Multi-stage stochastic scenario tree

The multi-stage stochastic linear program with recourse and tree structure can be formulated as

$$
\begin{aligned}
minZ = & \sum_{i \in \mathcal{V}} p_i c_i x_i \\
s.t. \ & A_1 x_1 = b_1, \\
& \sum_{j \in \mathcal{P}(a(i))} T_j x_j + A_i x_i = b_i, && i \in \mathcal{V} \\
& x_i \geq 0, && i \in \mathcal{V}.
\end{aligned}
$$

Methods have been applied to the multi-stage stochastic programming with block-separable recourse. Louveaux (1980) first performed the generalization of the primal approach for the multi-stage problem. Birge (1985) and Pereira and Pinto (1985) generalized the L-shaped method to the multi-stage problem as a nested decomposition method. Birge (1988) also extended the basic-factorization techniques to the multi-stage problem, but these techniques brought hieratical computational difficulties. Grinold (1976) formulated the multi-stage stochastic program as a finite Markov chain and as equivalent primal and dual optimization problems. Beale et al. (1986) performed the first order approach for the multi-stage program.

1.2 Robust Optimization

The robust optimization is also an approach to address parameter uncertainty in the optimization model. Unlike stochastic programming, it does not assume the uncertain parameters are random variables with known distributions, rather, its parameters are considered in uncertain sets or uncertain intervals. Robust optimization looks for the feasible solution for all possible values of unknown parameters, normally the optimal solution of the best worst case under data uncertainty. Currently, most studies are about single stage and two-stage robust optimization.

The research on robust optimization has recently received renewed attention. Kouvelis and Yu (1997) proposed a framework for robust discrete optimization, which seeks to find a solution that minimizes the worst case performance under a set of scenarios for the data. Their framework is a scenario based model.

They let the uncertainty data set be Ω, which is a scenario uncertainty set. Let x be the decision variable and $D(\xi)$ be the instance of the feasible region that corresponds to the uncertain parameter $\xi \in \Omega$. The function $Q(x, \xi)$ evaluates the feasible solution x. The robust optimization problem is designed to find the optimal solution x and the corresponding ξ such that

$$z = \min_{x} \max_{\xi \in \Omega} Q(x, \xi). \tag{1–3}$$

Ben-Tal and Nemirovski 1998, 2000 proposed the following framework on robust optimization:

$$\min_{x} \max_{D_0 \in \Omega_0} f_0(x, D_0)$$
$$s.t. \;\; f(x, D_i) \geq 0, \qquad i = 0, \cdots, T, D_i \in \Omega_i,$$

where $\Omega_i, i = 0, \cdots, T$, are the given uncertainty sets. Compared with Kouvelis and Yu (1997), their uncertainty set is more general. They showed that under the assumption that the set Ω_i is ellipsoids with the form

$$\Omega = \{D : \exists y, D = D^0 + \sum_{j \in N} \triangle D^j y_j, \|y\| \leq Y\}. \tag{1–4}$$

The robust counterparts of convex optimization problems are either exact or approximated tractable problems. But under the ellipsoid uncertainty, the robust counterpart of an LP becomes a nonlinear program. Under the same framework, Ben-Tal et al. (2004) studied the two-stage robust linear programming under the name *adjustable robust linear programming*. They presented that two-stage robust linear programming is computationally intractable and proposed a tractable alternative approach, referred to as *affinely-adjustable robust linear programming*.

Soyster (1973) proposed a linear robust optimization model to construct a solution that is feasible for all data that belong in a convex set. Bertsimas et al. (2004) and Bertsimas and Sim (2004) considered LPs whose coefficients of the objective function and constraints are assumed to be in uncertain intervals. Their approach retained the advantage of the linear framework of Soyster (1973) and protested constraints against violation. Bertsimas and Simchi-Levi (1996), Bertsimas and Sim (2003), and Bertsimas and Thiele (2006) applied their robust approach to address data uncertainty in discrete optimization. The perspective areas are vehicle routing, network, and inventory problems. They showed their robust approach is tractable in the above areas.

Bienstock and Özbay (2008) discussed the relaxation of the two-stage robust lot-sizing problem with basestock, and described a bender decomposition based algorithm for robust linear optimization. Atamtürk and Zhang (2007) described a two-stage robust optimization approach for solving network flow and design problems with uncertain demand. Two-stage robust lot-sizing with uncertain demand is an application for their approach. Most of these works are concentrated on the two-stage robust optimization.

1.3 Mixed-integer Linear Programming

The general form of the mixed-integer linear programming (MIP) is

$$\begin{aligned} \min\ & cx + hy \\ s.t.\ & Ax + Gy \leq b, \\ & x \geq 0, y \geq 0 \text{ and integer}, \end{aligned} \tag{1–5}$$

where $A \in \mathbb{R}^{m\times n}$, $G \in \mathbb{R}^{m\times p}$, $c \in \mathbb{R}^n$ and $h \in \mathbb{R}^p$, decision variable $x \in \mathbb{R}^n$, integer variable $y \in \mathbb{R}^p$. Let $S = \{(x, y) : Ax + Gy \geq b, x \geq 0, y \geq 0 \text{ integer}\}$ represent the feasible region of MIP. When x is also integral, we have a special case of MIP, an integer programming (IP). Nemhauser and Wolsey (1999) and Wolsey (1998) are comprehensive references for MIP.

Some MIP problems can be solved in polynomial time, such as the uncapacitated lot-sizing problem, the shortest path problem, the max flow problem, and the assignment problem. To date, no one has found a polynomial algorithm for these MIP problems, such as the 0-1 knapsack problem, the set covering problem, the traveling salesman problem, and the uncapacitated facility location problem. In the following section, we present useful results on polyhedral theory and algorithms for MIP.

1.3.1 Polyhedral Theory

First,we introduce the definitions of polyhedron and convex hull.

Definition 1. *The convex hull of S, denoted by $conv(S)$, is the set of all points that are convex combination of points in S.*

Definition 2. *A polyhedron $P \subseteq \mathbb{R}^n$ is the set of points that satisfy a finite number of linear inequalities; that is, $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, where (A, b) is an $m \times (n+1)$ matrix.*

Definition 3. *A bounded polyhedron is called a polytope.*

Definition 4. *F is a face of P, if (π, π_0) is a valid inequality for P, and $F = \{x \in P : \pi x = \pi_0\}$. A face F of P is a facet of P if $dim(F) = dim(P) - 1$.*

In essence, every polyhedron P can be described either by listing its facet-defining inequalities, or by its extreme points and extreme rays in the following theorems. Note that every polytope can be described by only its extreme points.

Theorem 1.1. *(Nemhauser and Wolsey 1999) A full-dimensional polyhedron P has a unique minimal representation by a finite set of linear inequalities.*

Theorem 1.2. *(Minkowski's Theorem) If $P \neq \emptyset$, then*

$$P = \{x \in \mathbb{R}^n : x = \sum_{k \in K} \lambda_k x^k + \sum_{j \in J} \mu_j r^j, \sum_{k \in K} \lambda_k = 1, \lambda_k \geq 0, \text{ for } k \in K, \mu_j \geq 0, \text{ for } j \in J.\},$$

where $(x^k)_{k \in K}$ is the set of extreme points of P and $(r^j)_{j \in J}$ is the set of extreme rays of P.

Theorem 1.3. *(Nemhauser and Wolsey 1999) The projection of a polyhedron is a polyhedron.*

Now, we show that the integer program can be reduced to be a linear program by the following theorems.

Theorem 1.4. *(Nemhauser and Wolsey 1999) If $S = \{x : Ax \leq b, x \in \mathbb{R}^n\}$, where (A, b) is an integer $m \times (n+1)$ matrix, then $conv(S)$ is a rational polyhedron.*

Theorem 1.5. *(Nemhauser and Wolsey 1999) IP is either infeasible or unbounded or has an optimal solution.*

The following definition and proposition show that there is a type of polyhedron having only integral extreme points.

Definition 5. *A nonempty polyhedron $P \subseteq \mathbb{R}^n$ is integral if each of its nonempty faces contains integral points.*

Proposition 1.1. *The following statements are equivalent:*

1. *P is integral.*

2. *LP has an integral optimal solution for all $c \in \mathbb{R}^n$ for which it has an optimal solution.*

3. *LP has an integral optimal solution for all $c \in \mathbb{Z}^n$ for which it has an optimal solution.*

We describe a class of matrices for which the integrality of LP holds with integral b.

Definition 6. *An $m \times n$ integral matrix A is totally unimodular (TU) if the determinant of each square submatrix of A is equal to 0, 1, or -1.*

Proposition 1.2. *If A is TU, then $P(b) = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ is integral for all $b \in Z^n$ for which it is not empty.*

1.3.2 The Branch-and-Cut Algorithm

First, we introduce the branch-and-bound algorithm. Let S represent the feasible region of MIP problem (1–5). The LP relaxation of MIP is $\min \{cx + hy, Ax + Gy \leq b, x \geq 0, y \geq 0\}$. Let P represent the feasible region of this LP problem.

Definition 7. *The collection $\{S^i : i = 1, \cdots, k\}$ is called a division of S if $\cup_{i=1}^k S^i = S$.*

Let L be a collection of MIP^i with $MIP^i = \min \{cx + hy, (x, y) \in S^i\}$, where $S^i \subseteq S$. The general branch-and-bound algorithm for mixed-integer programming can be described as follows:

The branch-and-bound algorithm (Nemhauser and Wolsey 1999)

Step 1 [Initialization]: $L = \{MIP\}$, $\underline{z}^0 = +\infty$, and $\overline{z}_{MIP} = +\infty$.

Step 2 [Termination test]: If $L = \emptyset$, then the solution (x^0, y^0) with objective value $\overline{z}_{MIP} = cx^0 + hy^0$ is optimal.

Step 3 [Problem selection and relaxation]: Select and delete problem MIP^i from L. Solve its relaxation RP^i. Let z_R^i be the optimal value of the relaxation and let (x_R^i, y_R^i) be an optimal solution if one exists.

Step 4 [Pruning]:

a. If $z_R^i \geq \overline{z}_{MIP}$, go to Step 2.

b. If $(x_R^i, y_R^i) \notin S^i$, go to Step 5.

c. If $(x_R^i, y_R^i) \in S^i$ and $cx_R^i + hy_R^i \leq \overline{z}_{MIP}$, let $\overline{z}_{MIP} = cx_R^i + hy_R^i$. Delete from L all problems with $\underline{z}^i \leq \overline{z}_{MIP}$. If $cx_R^i + hy_R^i = \overline{z}_{MIP}$, go to Step 2; Otherwise, go to Step 5.

Step 5 [Division]: Let $\{S^{ij}, j = 1, \cdots, k\}$ be a division of S^i. Add problems $\{MIP^{ij}, j = 1, \cdots, k\}$ to L, where $\underline{z}^{ij} = z_R^i$ for $j = 1, \cdots, k$. Go to Step 2.

The branch-and-cut algorithm is a generalization of the branch-and-bound algorithm. In Step 4, instead of going to Step 5, then branch-and-cut algorithm finds a valid inequality $a^1x + a^2y \leq a^0$ of MIP^i, such that $a^1x_R^i + a^2y_R^i > a^0$. After adding this inequality to LP relaxation RP^i, it is re-solved and and then goes back to Step 4.

In general, the number of nodes in the branch-and-bound enumeration tree is exponential in terms of the problem size, even for the integer program with binary variables. Almost all general MIP codes use a branch-and-bound algorithm with LP relaxation. The LP relaxation in the branch-and-bound algorithm provides the lower bound of the corresponding MIP. By adding cuts, the branch-and-cut algorithm improves the lower bound and reduces the number of branching nodes. In practice, the branch-and-cut algorithm normally performs better than the branch-and-bound algorithm in terms of computational time.

In this dissertation, we generate the customized branch-and-bound algorithm for lot-sizing with disruption in Chapter 2. We explore the polyhedral structures for stochastic lot-sizing problems in Chapters 4 and 5.

1.4 The Lot-sizing Problem

The mixed-integer programming formulation for the single item, uncapacitated lot-sizing problem (ULS) is (cf. Nemhauser and Wolsey (1999)):

$$
\begin{aligned}
\min \ & \sum_{i=1}^{T} (\alpha_i x_i + f_i y_i + h_i s_i) & \\
s.t. \ & s_{i-1} + x_i = d_i + s_i & i = 0, \cdots, T \\
& x_i \leq M y_i & i = 0, \cdots, T \\
& x_i, s_i \geq 0, \ y_i \in \{0, 1\} & i = 0, \cdots, T,
\end{aligned}
$$

where x_i represents the production quantity in period i, s_i represents the inventory amount at the end of period i, and y_i indicates the setup decision at period i. Parameters α_i, f_i, h_i, and d_i represent the production cost, setup cost, inventory cost, and demand in time period i, respectively. Without loss of generality, we can assume $s_0 = 0$ and tighten $M_i = \sum_{j=i}^{T} d_i$.

The study of lot-sizing problems can be traced back to Wagner and Whitin (1958) in which a polynomial time dynamic programming algorithm was developed to get an optimal solution of the problem. Later on, there is significant research progress in solving this class of problems. To solve ULS to optimality, dynamic programming, reformulation, and polyhedron study have been proposed. As a polynomially solvable approach, dynamic programming applied in ULS was first introduced by Wagner and Whitin (1958). They proposed an $\mathcal{O}(n^2)$ dynamic algorithm to solve ULS. Federgrun and Tsur (1991), Wagelmans et al. (1992), and Aggarwal and Park (1993) independently obtained $\mathcal{O}(n \log n)$ implementations of the Wagner and Whitin (1958) algorithm. Wagelmans et al. (1992) introduced the definition of the Wagner-Whitin costs, i.e., $\alpha_i + h'_i \geq \alpha_{i+1}$, $1 \leq i \leq T-1$, where α_i and h'_i are the unit production and inventory costs for time period i, and implemented an $\mathcal{O}(T)$ time dynamic programming algorithm.

The classic extended reformulation of the lot-sizing problem are the facility location, the shortest path, and the multicommodity network flow formulations. Krarup and Bilde (1977) first introduced the facility location reformulation of the lot-sizing problems. Martin (1987) developed a reformulation technique to transform ULS to a shortest path formulation. Radin and Choe (1979) provided the multicommodity network flow reformulation.

The compact description of the convex hull of all feasible solutions for ULS has been derived through different approaches. Barany et al. (1984b) introduced the well-known (ℓ, S) inequalities

$$\sum_{i \in S} x_i + \sum_{i \in \mathcal{I} \setminus S} d_{i\ell} y_\ell \geq d_{0\ell}, \tag{1–6}$$

together with $x_i \geq 0$ and $0 \leq y_i \leq 1$ to describe the convex hull of ULS, where $\ell \in \mathcal{I} = \{0, 1, \cdots, T\}$, $S \subseteq \mathcal{I}$, and $d_{ij} = \sum_{k=i}^{j} d_k$. In addition, for the uncapacitated lot-sizing problem with start-up costs, van Hoesel et al. (1994) presented an extended formulation and an $\mathcal{O}(T^2)$ time separation algorithm. For the Wagner-Whitin costs case, Pochet and Wolsey (1994) generated an extended formulation for ULS with $\mathcal{O}(T^2)$ constraints. Dynamic programming has been applied the uncapacitated lot-sizing problem and capacitated lot-sizing problems. For the capacitated lot-sizing problem, Baker et al. (1978) studied the dynamic lot-sizing prlbem with time-varying production capacity constraints. Fisher et al. (2001) proposed a method to mitigate end-effects in the dynamic lot-sizing by evaluating the end-of-horizon inventory level based on the classic EOQ model. Hartman et al. (2010) derived a new set of valid inequalities for the capacitated lot-sizing problem from the end-of-stage solutions of a dynamic programming algorithm.

1.5 Dissertation Outline

In Chapter 2 and Chapter 3, we study the lot-sizing problem with disruptions in the robust lot-sizing problem setting. In Chapter 2, we investigate cases in which a severe

event may happen such that the normal process will be disrupted. Our objective is to provide a robust schedule such that the total cost is minimized under the worst case scenario. In order to solve this problem more efficiently, we generate the customized branch-and-bound algorithm with optimality testing.

In Chapter 3, we extend the previous study to multiple disruption cases. We formulate the problem to be a multi-stage robust optimization problem. Our objective is to provide a multi-stage robust schedule such that the total cost is minimized under the worst case scenario. We explore the polyhedral structure of the lot-sizing problem with single disruption by generating facet-defining inequalities. The general reformulation scheme has been provided to transform the multi-stage robust optimization problem into an equivalent single stage program.

In Chapter 4 and Chapter 5, we consider a two-stage stochastic uncapacitated lot-sizing problem with deterministic demands and Wager-Whitin costs. In Chapter 4, we consider cases without backlogging. The optimal form of the inventory level has been explored. Based on the optimal form, we provide an extended formulation. Further, we provide the integral polyhedron description in the original space by projecting the extended formulation to the original space. In Chapter 5, we consider the case with backlogging. We investigate the relationship among the inventory, the setup, and the backlogging in the optimal solution and provide the optimal forms of inventory and backlogging levels. An extended formulation has been proposed.

In Chapter 6, we study the stochastic dynamic knapsack set which is naturally raised from the stochastic lot-sizing problem. First, we extend the results in the deterministic dynamic knapsack set to the stochastic setting, and generate valid path inequalities for the stochastic dynamic knapsack set. Second, the mixing and pairing schemes are applied for the stochastic dynamic knapsack set to generate valid tree inequalities. Third, the lifting schemes are adopted to the stochastic dynamic knapsack set. More valid inequalities are generated. Fourth, the parallel computing is applied to

the stochastic dynamic knapsack set to test the efficiency of generated inequalities for the stochastic capacitated lot-sizing problem.

Finally, in Chapter 7, we conclude this dissertation and propose future research directions.

CHAPTER 2
LOT-SIZING WITH DISRUPTION

2.1 Introduction

In practice, manufacturers expend lots of resources and efforts generating the long-term production plan. However, disruptions often occur and interrupt the regular planned production processes and cause demands in some time periods cannot be satisfied. Extra productions are required in order to cover the unfilled demands and bring extra reparation costs. The efficiency of new production plans is normally measured by the extra cost generated due to disruption. Previously, researchers have realized that a lot-sizing schedule has to be updated after the uncertain events. In previous research, Carlson et al. (1979) and Kazan et al. (2000), etc. have studied the nervousness of lot-sizing problems in which future demands are gradually acquired and the initial schedules have to be updated, which leads to extra production cost. Yang et al. (2005) studied how to recover the lot-sizing problems after the realization of disruptions. All these works are focused on the recovery planning after the occurrence of uncertain events and no recourse information is involved in the model to obtain the original schedule. As compared to the robust optimization approach, the above studies focus on studying the efficiency for the second-stage problem.

In this paper, we study the lot-sizing problem with potential disruptions as recourse via robust optimization to handle the uncertainty. We consider cases in which there is potentially one disruption, and in which the exact time of the disruption is uncertain. Once the disruption occurs, the corresponding recovery production follows to cover the unfilled demand and the extra production cost is generated. Our objective is to maintain the production planning with the consideration of a potential disruption during the production process and achieve the minimum objective value that considers the worst case scenario for the disruption.

The contribution of this paper lies in the fact that this paper proposes a robust production planning to address disruptions. As compared to the case in which the recovery production is performed after the occurrence of a disruption, our approach provides a more robust planning. To the best of our knowledge, our approach is the first to use the robust optimization approach to solve the lot-sizing problem with disruption. In the remaining part of this paper, we study different lot-sizing problems with disruptions. In Section 2.2, we study the lot-sizing problem with disruption and outsourcing. For this case, the lot-sizing problem with disruption can be reformulated as a two-stage $\min\max$ problem. A corresponding primal-dual approach can be constructed to solve the problem. In Section 3.4, we study the lot-sizing problem with disruption and backlogging. In Section 2.3.1, we study the non-setup cost case and develop a pre-processing algorithm to pre-calculate the parameters for the second stage problem. Then, we can formulate the problem as a single-stage problem. In Section 2.3.2, we develop a customized branch-and-bound algorithm to solve the case in which setup cost is considered. Finally, in Section 2.4, we provide the computational results that show the tractability and efficiency of our approaches.

2.2 Lot-sizing Problem with Disruption and Outsourcing

In this section, we consider outsourcing as the reparation approach for the lot-sizing problem with disruption. We assume that if disruption happens in time period i, then the production amount x_i originally scheduled in this time period will be purchased from other suppliers. That is, outsourcing happens in the same time period as the disruption happens, and the outsourcing amount is equal to x_i with unit outsourcing cost o_i.

Since after the disruption, we consider that outsourcing is an option besides production to satisfy demands, we also allow outsourcing as an option in he original schedule. For the case in which disruption is considered, all first-stage variables, the production levels, the original outsourcing levels, the inventory levels, and the setup decision, are not influenced by the disruption; but the second-stage decision determines

the time period when the disruption happens. If we assume the disruption time period to be t, then the corresponding formulation can be described as follows:

$$\min_{x,s,y,w} \left(\sum_{i=1}^{T} p_i x_i + h_i s_i + o_i w_i + f_i y_i + \max_t \left((o_t - p_t) x_t - f_t \right) \right)$$

$$(PO) \quad s.t. \qquad x_i + w_i + s_{i-1} = d_i + s_i, \quad i = 1, \cdots, T \tag{2–1}$$

$$x_i \leq M y_i, \quad i = 1, \cdots, T \tag{2–2}$$

$$x,\ w,\ s \in \mathbb{R}_+^T, \quad y \in \{0,1\}^T,$$

where w_i represents the outsourcing amount in time period i and $1 \leq t \leq T$. Constraint (3–2) indicates the inventory flow balance and constraint (3–3) indicates that production happens in the setup time period; no upper bound limit exists for the production amount. By introducing an artificial binary decision variable α_i to indicate whether disruption happens in time period i, PO can also be described as follows:

$$\min_{x,s,y,w} \left(\sum_{i=1}^{T} p_i x_i + h_i s_i + o_i w_i + f_i y_i + \max_{\alpha} \sum_{i=1}^{T} \alpha_i \left((o_i - p_i) x_i - f_i \right) \right)$$

$$(PO_1) \quad s.t. \qquad x_i + w_i + s_{i-1} = d_i + s_i, \quad i = 1, \cdots, T$$

$$x_i \leq M y_i, \quad i = 1, \cdots, T$$

$$\sum_{i=1}^{T} \alpha_i \leq 1,$$

$$x,\ w,\ s \in \mathbb{R}_+^T,\ y,\ \alpha \in \{0,1\}^T.$$

We can also relax α_i for each $i = 1, \cdots, T$ to be fractional and the following conclusion holds.

Proposition 2.1. *For the lot-sizing problem with a single disruption and outsourcing, the formulation can be simplified as the following two-stage min-max problem:*

$$\min_{x,s,y,w}\left(\sum_{i=1}^{T} p_i x_i + h_i s_i + o_i w_i + f_i y_i + \max_{\alpha}\sum_{i=1}^{T}\alpha_i\left((o_i - p_i)x_i - f_i\right)\right)$$

$$\begin{aligned}
(\mathrm{PO}_2)\ \ s.t. \qquad & x_i + w_i + s_{i-1} = d_i + s_i, && i = 1,\cdots,T\\
& x_i \le M y_i, && i = 1,\cdots,T\\
& \alpha_i \le 1, && i = 1,\cdots,T\\
& \sum_{i=1}^{T}\alpha_i \le 1,\\
& x,\ s,\ w,\ \alpha \in \mathbb{R}_+^T,\ y \in \{0,1\}^T.
\end{aligned}$$

Proof. We only need to prove that there exists an optimal solution for the above problem such that α^* is integral. We prove the claim by the contradiction method. For a given optimal solution $(x^*,\ s^*,\ y^*,\ w^*,\ \alpha^*)$, we can first observe that there exists an optimal solution in which $\alpha_i^* = 0$ if $(o_i - p_i)x_i^* - f_i \le 0$. Now we prove that for the given optimal solution $(x^*,\ s^*,\ y^*,\ w^*)$, there exists a corresponding integral optimal solution α^* for the following subproblem

$$\begin{aligned}
& \max_{\alpha} && \sum_{i=1}^{T}\alpha_i\left((o_i - p_i)x_i - f_i\right)\\
(\textit{PO-SUB})\ \ s.t. \qquad & && \alpha_i \le 1, && i = 1,\cdots,T\\
& && \sum_{i=1}^{T}\alpha_i \le 1,\\
& && \alpha \in \mathbb{R}_+^T.
\end{aligned}$$

If there exists a time period i in which $\alpha_i^* \in (0,1)$ and $\sum_{i=1}^{T}\alpha_i^* < 1$, then we can increase α_i^* to be 1, which leads to a larger objective value for the above subproblem. Contradiction.

If there exists a time period i in which $\alpha_i^* \in (0,1)$ and $\sum_{i=1}^T \alpha_i^* = 1$, then there must exist at least one more time period j in which $\alpha_j^* \in (0,1)$. If $(o_i - p_i)x_i^* - f_i = (o_j - p_j)x_j^* - f_j$, then we can increase α_i^* and decrease α_j^* such that either α_i^* or α_j^* becomes integral. Thus, we obtain a solution with a fewer number of fractional solutions. Following this same step, we can either obtain an integral solution with the same objective value or find a case in a certain step in which $(o_i - p_i)x_i^* - f_i \neq (o_j - p_j)x_j^* - f_j$. Under this scenario, without loss of generality, we can assume $(o_i - p_i)x_i^* - f_i > (o_j - p_j)x_j^* - f_j$. Then we can obtain a larger objective value for the subproblem if we increase α_i^* and decrease α_j^* by a small value $\epsilon > 0$. Contradiction.

Therefore, the original conclusion holds. □

By introducing new extra variable θ, we can formulate PO_2 as follows:

Theorem 2.1. *The lot-sizing problem with a single disruption and outsourcing can be transformed as the following single stage mixed-integer program:*

$$
\begin{array}{llll}
 & \min\limits_{x,s,y,w,q} & \sum\limits_{i=1}^{T}(p_i x_i + h_i s_i + o_i w_i + f_i y_i) + \theta & \\
\text{(DO)} & s.t. & (o_i - p_i)x_i - f_i \leq \theta, & i = 1, \cdots, T \\
 & & x_i + w_i + s_{i-1} = d_i + s_i, & i = 1, \cdots, T \\
 & & x_i \leq M y_i, & i = 1, \cdots, T \\
 & & x,\ s,\ w \in \mathbb{R}_+^T, & \theta \in \mathbb{R}_+,\ y \in \{0,1\}^T.
\end{array}
$$

From above analysis, we can solve the lot-sizing problem with a single disruption and outsourcing by a mixed-integer programming problem. We can also observe that the following (ℓ, S)-type inequalities (see, e.g., Barany et al. 1984a) are valid for *DO*.

2.3 Lot-sizing Problem with Disruption and Backlogging

In this section, we discuss the lot-sizing problem with disruption that utilizes backlogging as the reparation approach. We assume that the unit outsourcing cost is much larger than the unit backlogging cost and it is only utilized when no backlogging

can be obtained, e.g., in the pseudo time period $T + 1$. In the backlogging setting, the inventory and backlogging amounts after the time period in which disruption happens are changed. Therefore, we cannot adopt the same approach as the lot-sizing problem with disruption and outsourcing to solve this problem.

We introduce parameters $c_{i\ell}$ and auxiliary decision variables $x_{i\ell}$ as described in Pochet and Wolsey (1988). That is, parameter $c_{i\ell}$ represents the total cost of producing an item in time period i (except setup cost) to satisfy the demand in time period ℓ. For instance, for each $i \leq T$, we have $c_{i\ell} = p_i + \sum_{k=i}^{\ell} h_k$ if $\ell > i$, $c_{i\ell} = p_i + \sum_{k=\ell}^{i} b_k$ if $\ell < i$, and $c_{ii} = p_i$. If $i = T + 1$, then $c_{i\ell} = o_i + \sum_{k=\ell}^{T} b_k$. Problem parameters h_i and b_i represent the unit inventory and backlogging costs in time period i, respectively. Decision variable $x_{i\ell}$ represents the amount produced in time period i to satisfy the demand in time period ℓ. When a disruption happens in time period i, decision variable $x_{i\ell}$ represents the quantity of the unsatisfied demand in period ℓ. Therefore, the makeup production amount for period ℓ can be decided by the disruption period i and the corresponding original production amount used for this period $x_{i\ell}$. In order to provide the smallest cost to cover the unfilled demand in time period ℓ due to the disruption in time period i, we choose a period $q(\ell)$ that is setup and could provide the smallest cost to cover the unfilled demand in time period ℓ.

Note here once the first stage decision variables are given, for a given disruption period t, we can separately calculate the minimum makeup cost for each $x_{t\ell}$. Since there is no capacity, and the setup decision is provided in the first stage, the period which provides the minimum unit cost for the demand in period ℓ only depends on the time period ℓ. Then $\sum_{\ell=1}^{T} c_{q(\ell)\ell} x_{t\ell}$ is the smallest makeup cost.

Based on this idea and the application of the facitliy location reformulation, the lot-sizing problem with disruption and backlogging can be formulated as a two-stage

robust optimization problem as follows:

$$\min_{(x,y)} \sum_{i=1}^{T+1} f_i y_i + \sum_{i=1}^{T+1}\sum_{j=1}^{T} c_{ij}x_{ij} + \max_{1\le t\le T} \sum_{j=1}^{T} \min_{t<q(j)\le T+1 \text{ and } y_{q(j)}=1} \left(c_{q(j)j}x_{tj} - c_{tj}x_{tj} \right)$$

$$\text{(PR)} \quad s.t. \qquad \sum_{i=1}^{T+1} x_{ij} = d_j, \quad j = 1, \cdots, T \tag{2–3}$$

$$x_{ij} \le My_i, \qquad j = 1, \cdots, T;\ i = 1, \cdots, T+1 \tag{2–4}$$

$$x \in \mathbb{R}_+^{(T+1)\times T},\ y \in \{0,1\}^{T+1}.$$

The objective function is a two-stage optimization formulation. In the objective function, the second stage decision variable t is the index of time period. In the second stage, the minimum total makeup cost for a disruption can be determined by the sum of the minimum makeup cost for each x_{tj}, $j = 1, \cdots, T$. Therefore, time periods $q(j)$ are the decision variables which provide the minimum makeup cost for x_{tj}, $j = 1, \cdots, T$. The parameter $c_{q(j)j}$ is determined by $q(j)$, $t < q_{(j)} \le T+1$. Constraint (2–3) indicates that the demand in time period j should be satisfied. Constraint (2–4) indicates that for each time period, production happens only in the time period in which production is set up.

In this formulation, in the second stage, we only need to decide the time period in which the disruption happens and the time periods during which reparation productions happen. The reparation production quantity for each period j is equal to the first stage decision variable x_{tj}, where t is the disruption period.

2.3.1 Non-setup Cost Case

In this subsection, we consider a case without setup costs. Suppose disruption happens in time period t, with the second term of the objective function in PR, i.e., $\sum_{j=1}^{T} \min_{t<q(j)\le T+1 \text{ and } y_{q(j)}=1}(c_{q(j)j}x_{tj} - c_{tj}x_{tj})$, we have that $c_{q(j)j}$ is determined by the first and second stage variables. Since there is no setup cost, we propose the following *Pre-processing algorithm to determine* $c_{q(j)j}$. We have $m_{tj} = \min_{i>t} c_{ij}$ for each pair (t, j). By using backward induction and setting $m_{(t-1)j} = \min\{c_{(t-1)j}, m_{tj}\}$, we observe that m_{tj}

for all $t, j = 1, \cdots, T$ can be pre-calculated in $\mathcal{O}(T^2)$ time with $\mathcal{O}(T^2)$ storage space. With the Pre-processing algorithm, $c_{q(j)j} = m_{tj}$ when disruption happens in time period t.

Proposition 2.2. *For the non-setup cost lot-sizing problem with disruption and backlogging, suppose the disruption happens in time period t, the time period to cover the unfilled demand in time period ℓ is argmin$\{c_{i\ell},\ i > t\}$, and the corresponding minimum unit makeup cost is $m_{t\ell} = \min\{c_{i\ell},\ t < i \leq T+1\}$.*

With the pre-processing algorithm, we can pre-solve the second stage problem and get the following formulation:

$$\begin{aligned}
\min_x \quad & \sum_{i=1}^{T}\sum_{j=1}^{T} c_{ij}x_{ij} + \max_t \left(\sum_{j=1}^{T} m_{tj}x_{tj} - \sum_{j=1}^{T} c_{tj}x_{tj} \right) \\
s.t. \quad & \sum_{i=1}^{T} x_{ij} = d_j, && j = 1, \cdots, T \\
& x \in \mathbb{R}_+^{T\times T}.
\end{aligned}$$

With the above formulation, we can reformulate PR to be a single stage optimization problem which is presented in the following result.

Theorem 2.2. *The non-setup cost lot-sizing problem with disruption and backlogging has an equivalent linear programming formulation as follows:*

$$\begin{aligned}
& \min_x \quad \sum_{i=1}^{T}\sum_{j=1}^{T} c_{ij}x_{ij} + \theta \\
(\mathrm{PB}_1) \quad & s.t. \quad \sum_{j=1}^{T} m_{tj}x_{tj} - \sum_{j=1}^{T} c_{tj}x_{tj} \leq \theta, && t = 1, \cdots, T \\
& \sum_{i=1}^{T} x_{ij} = d_j, && t = 1, \cdots, T \\
& x \in \mathbb{R}_+^{T\times T},
\end{aligned}$$

where m_{tj} is calculated from a pre-processing algorithm.

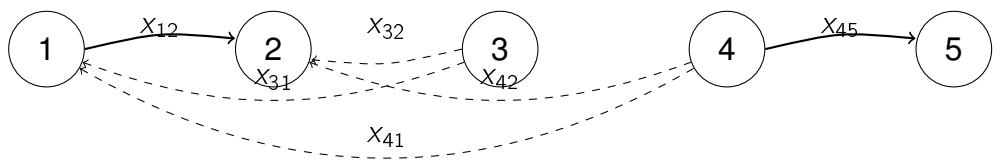


Figure 2-1. An example for the lot-sizing problem with disruption and backlogging

2.3.2 Setup Cost Case

When the setup cost is considered as shown in PR, we cannot determine the second stage parameters $c_{q(j)j}$ until the first stage decision variables $y_{q(j)}$ are decided, because the makeup production can only happen in the periods that are set up, i.e., $y_i = 1$. In this section, we introduce a reformulation of PR such that the problem can be formulated as a single stage problem and can be solved as a single stage mixed-integer programming problem.

We first introduce the following additional decision variables:

q_{tj}**:** The smallest total cost incurred in the second stage in order to satisfy the unfilled demand in time period j, due to the disruption in time period t.

z_{kj}^t**:** An auxiliary binary decision variable to indicate if we produce in time period k to satisfy the unfilled demand in time period j, where $t < k \leq T + 1$, when disruption happens in time period t. If yes, then $z_{kj}^t = 1$, otherwise, $z_{kj}^t = 0$.

θ**:** An auxiliary decision variable to represent the maximum increment cost after a disruption.

The main idea here is to formulate and study a single-stage mixed-integer programming problem, instead of solving a two-stage robust optimization problem. Before we describe the mathematical reformulation, Figure 2-1 shows an example to demonstrate the reformulation process. The figure shows a five period example. We assume periods 1, 3, and 4 are set up for production. Production in period 1 satisfies demands for itself and period 2, production in period 3 satisfies the demand for itself, and production in period 4 satisfies demands for periods 4 and 5.

As shown in Figure 2-1, suppose a disruption happens in time period 1. Then q_{11} and q_{12} represent the smallest makeup costs for the unsatisfied demand in periods 1 and 2, respectively. The makeup production may be from periods 3 or 4. Since variable θ represents the maximum increment cost after a disruption, we have $q_{11} + q_{12} - c_{11}x_{11} - c_{12}x_{12} \leq \theta$. Finally, time period $T+1$ serves as the pseudo-period to provide outsourcing. Thus, we have $y_{T+1} = 1$ with zero setup cost.

Theorem 2.3. *For the setup cost case, the lot-sizing problem with disruption and backlogging can be formulated as the following single-stage mixed-integer programming problem:*

$$
\begin{aligned}
\min_{x,y,\theta,q,z} \quad & \sum_{i=1}^{T+1} f_i y_i + \sum_{i=1}^{T+1}\sum_{j=1}^{T} c_{ij}x_{ij} + \theta \\
s.t. \quad & \sum_{j=1}^{T} q_{tj} - \sum_{j=1}^{T} c_{tj}x_{tj} \leq \theta, && t = 1, \cdots, T && (2\text{–}5) \\
& c_{kj}x_{tj} + M(z_{kj}^t - 1) \leq q_{tj}, && t, j = 1, \cdots, T;\ k = t, \cdots, T+1 && (2\text{–}6) \\
(\mathrm{PB}_2) \quad & \sum_{k=t+1}^{T} z_{kj}^t = 1, && t, j = 1, \cdots, T && (2\text{–}7) \\
& z_{kj}^t \leq y_k, && t, j = 1, \cdots, T;\ k = t, \cdots, T+1 && (2\text{–}8) \\
& \textit{(2–3), (2–4)}, \\
& x \in \mathbb{R}_+^{(T+1)\times T},\ y \in \{0,1\}^{T+1}, \\
& z \in \{0,1\}^{T\times T\times(T+1)},\ q \in \mathbb{R}_+^{T\times T},\ \theta \in \mathbb{R}_+.
\end{aligned}
$$

In the above formulation, artificial variable θ represents the total cost incurred in the second stage. Constraint (2–5) represents that θ should be the maximum increment cost due to the disruption at each time period $t = 1, \cdots, T$. Constraint (2–6) indicates the backlogging cost to fulfill the unfilled demand in time period j due to disruption in time period t. Constraint (2–7) represents that, due to the unlimited production capacity, there is only one later period needed to produce more to fulfill the unfilled demand for each time period j due to disruption in time period t. Constraint (2–8) represents that

the period provides reparation should be originally setup. Besides these, constraints (2–3) and (2–4) described in PR are also necessary.

2.3.2.1 A Branch-and-Bound Algorithm

Due to a large amount of binary decision variables z_{kj}^t in the formulation, it is very difficult to solve the problem into optimality. In order to efficiently solve PB_2, we generate a customized branch-and-bound algorithm. This algorithm guarantees to find an exact optimal solution and terminate in a finite number of steps.

In PB_2, there are two types of binary decision variables z and y. In our branch-and-bound algorithm, except leaf nodes, we apply the branch-and-bound procedure only for setup decision variable y and relax z to be fractional. At each leaf node in the resulting branch-and-bound tree generated by our procedure, decision variables y are integral. Then, we can solve a subproblem in which z is required to be integer and obtain a corresponding feasible solution.

In the remaining part of this section, we describe in detail our branching and searching strategies, lower and upper bounds in the branch-and-bound framework, and Benders decomposition framework for the optimality test. Before we describe the details, we first let L and V be the sets of leaf and total nodes in the enumeration tree, respectively. For each particular node $n \in V$, we let $y(n)$ and $d(n)$ be the solution of variable y and the depth of this node; let $a(n)$, $C(n)$ and $D(n)$ be the parent, children set and descendant set of node n; let $P(n)$ be the nodes on the path from the root node to node n.

2.3.2.2 Branching and Searching Strategies

In our branch-and-bound framework, we relax z to be fractional in our search process. Therefore, in our branch-and-bound process, we do not branch on z variables. If a y decision variable, for instance, y_i, becomes one at some node $n \in V$, it may become fractional when we solve the problem corresponding to a node which is a child of node n, and then become zero when we solve the problem corresponding to

a node which is a descendant of node n. Therefore, in order to get the global optimal solution, we need to consider all the combinations of integral y solutions. We will still branch the y variable when y variable is integral. In our process, we first branch the decision variable y_1 at the root node with depth 1. Then, in general, corresponding to each node with depth d, we will branch the decision variable y_d. Finally, we apply the depth first search, which allows us to find good feasible solutions at early stages during the branch-and-bound process.

2.3.2.3 The Lower and Upper Bounds

We first can observe the following conclusion:

Proposition 2.3. *In the branch-and-bound tree, corresponding to each node n, the optimal objective value of the linear programming relaxation for this node provides the lower bound for its descendant.*

Proof. Our branch-and-bound policy is not the same as the traditional one. For instance, we need to branch a y_i decision variable even if we get an integral y_i solution. However, corresponding to each node in the tree, we solve the linear programming relaxation and it still provides the lower bound for all its descendant. □

To obtain an upper bound, at each node in the branch-and-bound tree, if we have all y decision variables integral, then we can solve a subproblem corresponding to this integral y solution. As shown in PB_2, once y decision variables are fixed, we can write down a reformulation for the problem to decide the production quantity, the disruption period, and the reparation production periods in the second stage. Corresponding to each node $n \in V$ such that all y decision variables are integral in the linear programming relaxation solution, let $\mathcal{I}(n) = \{t = 1, \cdots, T : y_t = 1\}$. Each period $t \in \mathcal{I}(n)$ can be a candidate in which the disruption happens. Once a disruption happens in time period t, only the periods that are set up after time period t can provide extra production quantity to satisfy unfilled demands which are caused by the disruption in time period t. Corresponding to each pair (t, ℓ), where t represents the disruption

time period and ℓ represents the period in which the demand is unfilled, we can define $m_{t\ell} = \min\{c_{i\ell},\ i \in \mathcal{I}(n),\ t < i \leq T+1\}$.

$$\text{If } k = \text{argmin}\{c_{i\ell},\ i \in \mathcal{I}(n),\ t < i \leq T+1\}, \text{then } z^t_{k\ell} = 1;\ \text{otherwise } z^t_{k\ell} = 0. \tag{2–9}$$

Then, we can write down the following subproblem to obtain a feasible solution for PB_2, which is an upper bound for the problem.

$$\text{(PB-SUB)}\quad \begin{array}{lll} \min & \displaystyle\sum_{i=1}^{T+1}\left(\sum_{\ell=1}^{T} c_{i\ell}x_{i\ell} + f_i y_i\right) + \max_{t\in\mathcal{I}(n)}\left(\sum_{\ell=1}^{T} m_{t\ell}x_{t\ell} - \sum_{\ell=1}^{T} c_{t\ell}x_{t\ell}\right) & \\ s.t. & \displaystyle\sum_{i=1}^{T+1} x_{i\ell} = d_\ell, & \ell = 1,\cdots,T, \\ & x_{i\ell} \leq My_i, & \ell = 1,\cdots,T;\ i = 1,\cdots,T+1, \\ & x \in \mathbb{R}_+^{(T+1)\times T}. & \end{array}$$

The above formulation can be easily solved since it is a linear program. It is obvious that for each leaf node, we have y decision variables integral and an upper bound can be obtained by solving PB-SUB. Following a similar process as the traditional branch-and-bound procedure, we initialize the upper bound $UB = +\infty$ and the lower bound $LB = -\infty$. At each node $n \in V$, if not all y decision variables are integral by solving the linear program corresponding to this node, then we only obtain a lower bound $LB(n)$ for the descendants of node n. If $LB(n) > UB$, then we prune node n and its descendants. If all y decision variables are integral by solving the linear program corresponding to this node, then we can obtain both an upper bound $UB(n)$ and a lower bound $LB(n)$ for the problem. If $UB(n) < UB$, then we let $UB = UB(n)$. If $LB(n) > UB$, then we prune node n and its descendants. Otherwise, we continue branching and performing the depth first search. Finally, we will terminate at an optimal solution.

2.3.2.4 The Optimality Test

We apply the Benders' decomposition approach at each leaf node to test if we have obtained the optimal solution for the problem. In the Benders' decomposition framework,

we let (y, z) be the decision variables for the master problem and others be the decision variables for each slave problem. The master problem can be described as follows:

$$
\begin{aligned}
&\min_{y,z} \quad \eta \\
&s.t. \quad \eta \geq \sum_{t=1}^{T} f_t y_t + \sum_{t=1}^{T}\sum_{j=1}^{T}\sum_{k=t+1}^{T} M\mu_{kj}^{t}(1 - z_{kj}^{t}) + \sum_{t=1}^{T} M\nu_t y_t + \sum_{t=1}^{T} \omega_t d_t, \qquad (2\text{–}10) \\
(\textbf{Master}) \quad & z_{kj}^{t} \leq y_k, \qquad t, j = 1, \cdots, T;\ k = t, \cdots, T+1, \\
& \sum_{k=t+1}^{T} z_{kj}^{t} = 1, \qquad t, j = 1, \cdots, T, \\
& y \in \{0,1\}^{T+1},\ z \in \{0,1\}^{T\times T\times(T+1)},
\end{aligned}
$$

where inequality (2–10) is generated based on the following subproblem and μ, ν, and ω are the dual values corresponding to constraints (2–11), (2–12), and (2–13).

$$
\begin{aligned}
Z(y,\ z) = \min_{x,\theta,q} \quad & \sum_{i=1}^{T+1}\sum_{j=1}^{T} c_{ij}x_{ij} + \theta \\
s.t. \quad & \sum_{j=1}^{T} q_{tj} - \sum_{j=1}^{T} c_{tj}x_{tj} - \theta \leq 0, \qquad t = 1, \cdots, T, \\
(\textbf{Slave}) \quad & c_{kj}x_{tj} - q_{tj} \leq M(1 - z_{kj}^{t}), \quad t, j = 1, \cdots, T;\ k = t, \cdots, T+1, \qquad (2\text{–}11) \\
& x_{i\ell} \leq My_i, \qquad \ell = 1, \cdots, T;\ i = 1, \cdots, T+1, \qquad (2\text{–}12) \\
& \sum_{i=1}^{T+1} x_{i\ell} = d_\ell, \qquad \ell = 1, \cdots, T, \qquad (2\text{–}13) \\
& x \in \mathbb{R}_{+}^{(T+1)\times T},\ q \in \mathbb{R}_{+}^{T\times T},\ \theta \in \mathbb{R}_{+}.
\end{aligned}
$$

In our optimality test procedure, for a given leaf node in which we have y decision variables integral, we can obtain the corresponding integral z values. For instance, suppose $(\overline{y},\ \overline{z})$ is the solution corresponding to a leaf node. By solving PB-SUB, we will obtain the optimal objective value $Z(\overline{y},\ \overline{z})$ for the slave problem. Meanwhile, we will obtain the dual values corresponding to constraints (2–11), (2–12), and (2–13). Thus, we will obtain a corresponding $\overline{\eta}$ in the master problem. Note here $y_{T+1} =$

1 is predefined. Therefore, (Slave) is always feasible and bounded for any given $\overline{y}$ solution and we do not need to put extreme ray constraints in the Master problem for the optimality test. The optimality test for Benders' decomposition of PB_2 is

$$Z(\overline{y},\ \overline{z}) + \sum_{t=1}^{T} f_t \overline{y}_t \leq \overline{\eta}. \tag{2–14}$$

If inequality (2–14) is satisfied, then $(\overline{y},\ \overline{z},\ x)$ is an optimal solution of PB_2. Since we only need to test if the current integral solution y can lead to an optimal solution for the original problem, we do not need to solve the master problem. Thus, we only need to solve an extra linear programming problem to obtain an updated η each time. The details are shown as follows:

Algorithm: The optimality test

Step 0: Initialize $\eta = +\infty$ and $U = \emptyset$, where U stores the dual solution of Slaves.

Step 1: At the kth leaf node in the branch-and-bound tree, denoted as node n_k, we perform the following operations:

Step 1.1: Based on $y(n_k)$, we obtain $z(n_k)$ as shown in (2–9).

Step 1.2: Solve the Slave problem to obtain the corresponding objective value $\bar{Z}$, and add the corresponding dual values to set U.

Step 1.3: Let $\eta = \max_{(\mu,\ \nu,\ \omega)\in U}\{\sum_{t=1}^{T} f_t y_t(n_k) + \sum_{t=1}^{T}\sum_{j=1}^{T}\sum_{k=t+1}^{T} M\mu_{kj}^{t}(1 - z_{kj}^{t}(n_k)) + \sum_{t=1}^{T} M\nu_t y_t(n_k) + \sum_{t=1}^{T}\omega_t d_t\}$.

Step 2: Optimality test: If $\bar{Z} + \sum_{t=1}^{T} f_t y_t(n_k) \leq \eta$, then $(x(n_k), y(n_k), q(n_k), z(n_k), \theta(n_k))$ is the optimal solution for the original problem, stop. Otherwise, wait until the next leaf node and go to Step 1.

Since the optimal y solution must exist at some leaf nodes, the optimality test will be satisfied at a leaf node in the branch-and-bound framework. Thus, the algorithm will terminate in finite steps.

Note here, the Bender's decomposition formulation can be used for both solving the problem from scratch and serving as the optimality test. In the later computational experiment, we apply the Bender's decomposition for both purposes.

2.4 Computational Results

In this section, we present the computational results to demonstrate the computational tractability of solution approaches we studied for different cases of the robust lot-sizing problems with disruption. All computational experiments were carried out on a Linux workstation with a Pentium Dual 2.8G processor and 6G RAM. We used CPLEX 10.1 Callable Library to implement our algorithms, and run the reformulated models.

2.4.1 Instance Generation

We generate instances based on different ratios of setup cost f_i to unit production cost p_i, and different time horizons T. For the instances, we set the time horizon $T = 10, 20, 30, 40$ and 50, and the ratios of setup cost to unit production cost $f/p = 10$, 20, 30, and 40, respectively. There are 20 combinations in total.

For the cases with setup costs, corresponding to each of the combinations of T and f/p, we generate random instances in which the unit production cost and the setup cost are uniformly distributed in the intervals as shown in Table 2-1.

For the cases without setup costs, corresponding to each of the combinations of T and f/p, we generate random instances in which the unit production cost is uniformly distributed in the same interval as for the cases with setup costs shown in Table 3-1.

Table 2-1. Parameter setting

	unit production cost p_i	setup cost f_i
ratio 10	[50, 100]	[500, 1000]
ratio 20	[50, 100]	[1000, 2000]
ratio 30	[40, 60]	[1000, 2000]
ratio 40	[40, 60]	[500, 1000]

We also set demand d_i, unit inventory cost h_i, and unit backlogging cost b_i uniformly distributed in the intervals $[500, 1000]$, $[5, 10]$, and $[10, 20]$, respectively. Finally, we let unit outsourcing cost be fixed at 200.

2.4.2 Heuristic: Maximum Pick

We first use maximum pick, a simple heuristic, to serve as a base for comparison with other models. Based on the optimal solution of the single item lot-sizing problem without disruption, the maximum pick heuristic picks a period, in which the outsourcing brings the largest increment of the total cost, to do outsourcing. The detailed maximum pick heuristic is listed as follows:

Heuristics: Maximum Pick (MPH)

Step 1: Solve the single item lot-sizing problem. Let x^* be the optimal solution and z^* be the corresponding objective value.

Step 2: Sort $(o_t - p_t)x_t^*,\ 1 \leq t \leq T$ from the largest to the smallest, and record them as mp_1, mp_2, ..., mp_T.

Step 3: Use $z^* + mp_1$ as the increment cost of the total cost due to the disruption.

2.4.3 Lot-sizing with Disruption and Outsourcing

We test the cases using outsourcing to recover unfilled demand. For this case, as described in section 4.1, we test 20 combinations in which the time horizon $T = 10, 20, 30, 40,$ and 50, and the ratio $f/p = 10, 20, 30,$ and 40, respectively. The computational results are shown in Tables 2-2. For each of the 20 combinations, we report the average values of 5 random instances. We report 1) the optimal objective value of the lot-sizing problem without disruptions, denoted as "SLS", 2) the objective value obtained by maximum pick heuristic, denoted as "MPH", 3) the objective value for the robust optimization formulation obtained by using CPLEX to solve the dual formulation (DO) for the outsourcing case and 4) the gap between (DO) and (MPH), denoted as "GAP(M::R)=(Obj_{MPH}-Obj_{DO})/(Obj_{DO})," and 5) the gap between (SLS) and (DO), denoted as "GAP(R::S)= (Obj_{DO}-Obj_{SLS})/(Obj_{SLS})." Compared with the maximum

pick heuristics, the average gap is 16.7%. That means the total cost can be reduced by 16.7% by applying robust optimization approach (DO). Compared with the uncapacitated lot sizing problem without disruptions, the average gap between (SLS) and (DO) is 18.9%. That means total cost increases 18.9% by considering the disruption. We can also observe that as T increases, the average gap between SLS and DO decreases.

2.4.4 Lot-sizing with Disruption and Backlogging

Finally, we test the cases using backlogging to recover unfilled demand. We perform computational experiments for cases both with and without setup cost.

For the case without setup cost, we perform the computational experiments based on the pre-processing algorithm and reformulation (PB_1) described in Section 2.3.1. We test 10 combinations such that the ratios between the setup cost and the production cost equal to 10 and 30, and the planning horizons are 10, 20, 30, 40 and 50 respectively. Corresponding to each combination, we report the values of "SLS," "MPH," "NoSetup," and "GAP(M::R)" as described in the outsourcing case.

The pre-processing algorithm needs $\mathcal{O}(T^2)$ time. Formulation (PB_1) is a linear program with $\mathcal{O}(T^2)$ variables and $\mathcal{O}(T)$ constraints. Therefore, the problems for this case can be solved in short time and we do not show the computational time. Instead, we only report the cost savings by applying robust optimization approach as compared to the maximum pick heuristic, the case without considering the disruption during the planning process. The comparison is shown on Table 2-3, and in average, the cost saving is 27.8%.

For the case with setup cost, we compare different solution approaches that include solving (PB_2) described in Section 2.3.2 directly by default CPLEX, the proposed branch-and-bound algorithm, and the Benders decomposition approach. We set the time limit to be 2 hours and test the combinations that ratio=10, 20, 30 and 40, and the time period $T = 10, 20$ and 30, respectively. The computational results are shown in Table 2-4. We compare three different approaches that include the default CPLEX MIP

solver approach, our proposed branch-and-bound (BB) algorithm, and the Benders' decomposition algorithm. If the optimal solution cannot be obtained within the time limit, we report the final optimality gap. Otherwise, if the optimality gap is zero, we report the computational time in the parenthesis, in terms of seconds. From Table 2-4, we can observe that our proposed BB method performs much better than default CPLEX. For $T = 10$ and 20 cases, no instances can be solved into optimality within the given time limit by the default CPLEX. The final optimality gaps provided by the default CPLEX are in the interval $[1.13\%, 5.39\%]$. Our proposed branch-and-bound algorithm can solve all instances into optimality for $T = 10$ and $T = 20$ cases: for $T = 10$ cases, most instances can be solved into optimality within 5 minutes by our BB approach; for $T = 20$ cases, most instances can be solved into optimality within 1 hour. For $T = 30$ cases, neither the default CPLEX nor our proposed BB algorithm can solve the problems into optimality within 2 hours. However, our proposed BB algorithm obtains smaller optimality gaps as compared to the default CPLEX. For the Benders' decomposition approach to solve (PB_2) directly instead of serving as optimality test, we found that even for 10-period instances, no instances can be solved into optimality within 24 hours. Therefore, the computational results indicate that our proposed solution approach is the best among all three approaches.

Table 2-2. Lot-sizing with disruption: outsourcing

		$T = 10$	$T = 20$	$T = 30$	$T = 40$	$T = 50$
ratio=10	SLS	5.00×10^5	1.05×10^6	1.54×10^6	2.06×10^6	2.50×10^6
	MPH	8.44×10^5	1.51×10^6	2.04×10^6	2.61×10^6	2.97×10^6
	DO	6.67×10^5	1.25×10^6	1.78×10^6	2.35×10^6	2.79×10^6
	GAP(M::R)	26.5%	20.1%	14.6%	11.2%	6.6%
	GAP(R::S)	33.3%	19.3%	15.9%	14.2%	11.5%
ratio=20	SLS	4.99×10^5	1.06×10^6	1.52×10^6	2.01×10^6	2.47×10^6
	MPH	7.96×10^5	1.48×10^6	2.05×10^6	2.60×10^6	3.09×10^6
	DO	6.39×10^6	1.27×10^6	1.77×10^6	2.31×10^6	2.79×10^6
	GAP(M::R)	24.5%	16.3%	15.8%	12.3%	10.7%
	GAP(R::S)	28.0%	20.3%	16.3%	14.8%	12.6%
ratio=30	SLS	3.88×10^5	7.41×10^5	1.12×10^6	1.46×10^6	1.88×10^6
	MPH	6.85×10^5	1.06×10^6	1.42×10^6	1.81×10^6	2.27×10^6
	DO	5.26×10^5	8.98×10^5	1.29×10^6	1.64×10^6	2.08×10^6
	GAP(M::R)	30.2%	17.8%	9.9%	10.5%	9.1%
	GAP(R::S)	35.7%	21.2%	14.8%	12.5%	10.4%
ratio=40	SLS	3.68×10^5	7.81×10^5	1.15×10^6	1.52×10^6	1.92×10^6
	MPH	7.28×10^5	1.12×10^6	1.53×10^6	1.91×10^6	2.29×10^6
	DO	5.17×10^5	9.45×10^5	1.31×10^6	1.69×10^6	2.10×10^6
	GAP(M::R)	40.9%	18.8%	16.6%	12.9%	8.8%
	GAP(R::S)	40.5%	20.9%	14.4%	11.7%	9.8%

Table 2-3. Lot-sizing with disruption: backlogging and non-setup cost

		$T = 10$	$T = 20$	$T = 30$	$T = 40$	$T = 50$
ratio=10	SLS	5.03×10^5	9.86×10^5	1.56×10^6	2.08×10^6	2.44×10^6
	MPH	8.23×10^5	1.35×10^6	2.01×10^6	2.54×10^6	2.95×10^6
	NoSetup	5.39×10^5	1.02×10^6	1.61×10^6	2.12×10^6	2.47×10^6
	Gap(M::N)	34.6%	24.2%	19.9%	16.5%	16.1%
ratio=30	SLS	3.49×10^5	7.04×10^5	1.08×10^6	1.44×10^6	1.85×10^6
	MPH	6.26×10^5	1.03×10^6	1.35×10^6	1.77×10^6	2.19×10^6
	NoSetup	3.70×10^5	7.35×10^5	1.11×10^6	1.47×10^6	1.88×10^6
	Gap(M::N)	69.1%	40.3%	21.7%	20.5%	16.1%

Table 2-4. Lot-sizing with disruption: backlogging and branch-and-bound algorithm

			T=10	T=20	T=30
ratio=10	Gap(Op)	CPLEX	1.42%	4.39%	4.42%
		BB	0(371)	0(3974)	2.75%
ratio=20	Gap(Op)	CPLEX	2.76%	3.79%	4.71%
		BB	0(322)	0(3400)	2.66%
ratio=30	Gap(Op)	CPLEX	1.13%	1.59%	2.19%
		BB	0(300)	0(2959)	1.68%
ratio=40	Gap(Op)	CPLEX	1.21%	2.33%	2.43%
		BB	0(365)	0(2918)	1.79%

CHAPTER 3
MULTI-STAGE ROBUST LOT-SIZING WITH DISRUPTIONS

3.1 Introduction

Uncertainty is a natural property of disruptions in production planning problems. The exact times when disruptions happen are unpredictable. In this paper, we study the multi-stage robust lot-sizing problem as an example to analyze solution approaches for multi-stage robust integer programming problems. We consider the lot-sizing problem in which the disruption occurrence is uncertain. Once a disruption occurs, subsequent recovery productions follow to cover the unfilled demands with their corresponding recovery production costs. Our objective is to maintain the production planning with disruptions. Meanwhile, the production plan still can provide the minimal production cost under the worst case scenario (the largest production cost growths). In general, a multi-stage robust mixed-integer programming setting can be described as follows:

$$\begin{aligned} & \min_{(x_1,y_1)\in\mathcal{M}_1} [f_1(x_1,y_1) + \max_{\omega_1\in\Omega_1} [\min_{(x_2,y_2)\in\mathcal{M}_2(x_1,y_1,\omega_1)} f_2(x_1,y_1,\omega_1,x_2,y_2) + \dots \\ + \quad & \max_{\omega_{(T-1)}\in\Omega_{(T-1)}} [\min_{(x_T,y_T)\in\mathcal{M}_T(x_1,y_1,\dots,\omega_{(T-1)})} f_T(x_1,y_1,\omega_1,\dots,x_{(T-1)},y_{(T-1)},\omega_{(T-1)},x_T,y_T)]]], \end{aligned} \tag{3–1}$$

where (x_t, y_t) represents the decision in stage t, and Ω_t represents the uncertainty set in stage t. Decision variables in each stage t will satisfy constraints that describe the feasible region in stage t, denoted as $\mathcal{M}_t(x_1, y_1, x_2, y_2, \dots, \omega_{(t-1)})$. For the multi-stage robust lot-sizing problems, the above general multi-stage robust integer programming formulation can be applied to formulate lot-sizing with uncertain disruptions. The objective is to minimize the total cost under the worst case scenario.

The contributions of this chapter are threefold. First, this paper proposes a robust production planning to address disruptions. As compared to the case in which the recovery production is performed after the occurrence of disruptions, we provide more robust initial production planning. Second, this paper proposes a robust production planning to address multiple disruptions. As compared to the case in which there is only one disruption, the multiple disruptions case is much more challenging to

solve. Third, this paper studies solution approaches to solve both two-stage and multi-stage robust integer programming problems. In particular, this paper studies the polyhedral structures of the corresponding models for the lot-sizing problem with a single disruption and shows the computational effectiveness of our approaches. To the best of our knowledge, all these three aspects have never been studied before.

In the remaining part of this paper, we study different multi-stage robust lot-sizing problems which are tractable. In Section 3.2, we present the motivation, introduce our robust lot-sizing problem, and develop a general formulation for the multi-stage robust lot-sizing problem. In Section 3.3, we study the robust lot-sizing problem with outsourcing. For this case, the multi-stage robust lot-sizing problem can be reformulated as a two-stage $\min\max$ problem. A corresponding primal-dual approach can be constructed to solve the problem. In Section 3.4, the robust lot-sizing problem with backlogging and a single disruption is studied. We formulate this problem as a two-stage robust mixed-integer program. We propose a reformulation rule to transfer this robust model to a mixed-integer program. For this problem, we study the polyhedral structure and generate the corresponding facet-defining inequalities based on the reformulation. In Section 3.5, we consider cases with and without setup cost of the robust lot-sizing problem with backlogging and multiple disruptions. We develop a deterministic equivalent formulation for the multi-stage robust optimization problem, based on the enumeration of the periods in which disruptions happen. Finally, in Section 3.6, we provide the computational results that show the tractability and efficiency of our approaches.

3.2 The General Formulation

For a single-item production planning process, when disruptions happen in given time periods, the scheduled production is terminated in these time periods. We adopt outsourcing and backlogging as approaches to accommodate the supply shortage. That is, unfilled demands due to disruption in certain time period can be backlogged by

increasing the production in later time periods. This approach is referred to as a robust lot-sizing problem with backlogging case. For this case, if there is only one disruption, then the scheduled productions after the disruption time period can be processed, and extra production amount can be introduced at a time period which is set up after the disruption time period to accommodate the supply shortage. If there are multiple disruptions, then there are multiple iterations in terms of disruptions and recoveries. For instance, after the first disruption, the production process after the disruption time period will be rescheduled to counteract the disruption. Similarly, when the second disruption happens, reparation production will happen again. Consequently, disruptions and the corresponding reparation productions will happen consecutively.

To formulate the robust lot-sizing problem with uncertain disruptions, we introduce a hidden disruption parameter $\xi \in \{0,\ 1\}^T$. If there is a disruption that happens in period $i \in \mathbb{I} = \{1, \ldots, T\}$, then $\xi_i = 1$; otherwise, $\xi_i = 0$. Accordingly, we denote the set of disruption periods $S_0 = \{i,\ \xi_i = 1,\ i \in \mathbb{I}\}$. We assume that the total number of disruption periods is β. We also let x^1, s^1 and y^1 represent the first stage production, stocking, and set-up decision variables, and x^k, s^k and y^k represent the corresponding variables after the $k-1^{th}$ disruption. Then, the the objective function of the multi-stage robust lot-sizing problem can be described as follows:

$$
\begin{aligned}
\min_{(x^1,y^1,s^1)} \quad & \Bigg\{ \sum_{i=1}^{T} (p_i x_i^1 + f_i y_i^1 + h_i s_i^1) + \max_{t_1 \in S_0} \min_{(x^2,s^2,y^2)} \Bigg\{ -\xi_{t_1} \psi_1(t_1, x, y) + \sum_{i=t_1+1}^{T} (\phi_i x_i^2 + h_i s_i^2) \\
& + \max_{t_2 \in S_0, t_2 > t_1} \min_{(x^3,s^3,y^3)} \Bigg\{ -\xi_{t_2} \psi_2(t_2, x, y, x^1, y^1) + \sum_{i=t_2+1}^{T} (\phi_i x_i^3 + h_i s_i^3) \\
& + \max_{t_3 \in S_0, t_3 > t_2} \min_{(x^4,s^4,y^4)} \Bigg\{ -\xi_{t_3} \psi_3(t_3, x, y, x^1, y^1, x^2, y^2) + \sum_{i=t_3+1}^{T} (\phi_i x_i^4 + h_i s_i^4) \\
& + \cdots\cdots \\
& + \max_{t_\beta \in S_0, t_\beta > t_{\beta-1}} \min_{(x^\beta,s^\beta,y^\beta)} \Bigg\{ -\xi_{t_\beta} \psi_\beta(t_\beta, x, y, \cdots, x^{\beta-1}, y^{\beta-1}) + \sum_{i=t_\beta+1}^{T} (\phi_i x_i^{\beta+1} + h_i s_i^{\beta+1}) \Bigg\} \Bigg\} \Bigg\} \Bigg\} \Bigg\} \Bigg\},
\end{aligned}
$$

where we generalize the cost parameter for the recovery production quantity to be the unit recovery production cost if backlogging is applied in time period i for each $i \in \mathbb{I}$. Besides this, in our objective setting, if the jth disruption happens in time period t_j, then the corresponding cost $\psi_j(t_j, x, y, x^1, y^1, \cdots, x^{j-1}, y^{j-1})$ is saved. For instance, if there is a disruption that happens in time period t, then the scheduled production cost cannot be processed, and the corresponding costs (including production, inventory, and backlogging costs) are saved. The exact formulation of saving cost function $\psi_i(\dots)$ and the constraints set depend on the recover approaches. In the remaining part of this paper, we study the robust lot-sizing problem with backlogging cases.

3.3 The Robust Lot-sizing Problem with Outsourcing

In this section, we study the multi-stage robust lot-sizing problem with outsourcing. For this case, we consider outsourcing as the reparation approach. We assume if disruption happens in a time period t, then the production amount x_t originally scheduled in this time period will be purchased from other suppliers. That is, outsourcing happens in the same time period as the disruption happens, and outsourcing amount is equal to x_t with unit outsourcing cost o_t.

Under this scenario, since we consider that outsourcing is also an option besides production to satisfy demands after the disruption, for the deterministic lot-sizing problem without disruptions, we will also have outsourcing option considered. For the case in which disruption is considered, the production amount, the original outsourcing amount, the inventory amount, and the setup decision are all first-stage decision variables and not influenced by the disruption. The only second-stage decision variable is the time period t when the disruption happens. If multiple disruptions happen, in each disruption time period, the outsourcing amount is equal to the reduced production quantity due to the disruption. Accordingly, we still only need to decide the periods when the disruptions happen. Therefore, the multi-stage robust lot-sizing problem with outsourcing can be transferred to be a two-stage robust optimization problem with

multiple disruptions happening in the second stage. In this case, if the jth disruption happens in time period t_j, the production costs in this time period can be saved. Thus, the lost function is formulated as $\psi_j(t_j, x, y) = p_{t_j} x_{t_j}$. If we assume the set of disruption periods to be S_0, a subset of $\mathbb{I}$, and the number of disruptions to be β, i.e., $|S_0| = \beta$. The corresponding formulation can be described as follows

$$\min_{x,s,y,w} \left(\sum_{i=1}^{T} p_i x_i + h_i s_i + o_i w_i + f_i y_i + \max_{S_0} \sum_{t \in S_0 \subseteq \mathbb{I}} ((o_t - p_t) x_t) \right)$$

$$s.t. \qquad x_i + w_i + s_{i-1} = d_i + s_i, \quad i \in \mathbb{I} \tag{3–2}$$

$$x_i \leq M y_i, \quad i \in \mathbb{I} \tag{3–3}$$

$$x,\ w,\ s \in \mathbb{R}_+^T, \quad y \in \{0,1\}^T,$$

where w_i represents the outsourcing amount in time period i. Constraint (3–2) indicates the inventory flow balance and constraint (3–3) indicates that production happens in the time period in which the production is set up and the production amount does not have an upper bound limit since M is a very large number. Let artificial binary decision variable α_i indicate if the disruption happens in time period i, the above formulation can also be described as follows:

$$\min_{x,s,y,w} \left(\sum_{i=1}^{T} p_i x_i + h_i s_i + o_i w_i + f_i y_i + \max_{\alpha} \sum_{i=1}^{T} \alpha_i \left((o_i - p_i) x_i \right) \right)$$

$$(RFLS) \quad s.t. \qquad x_i + w_i + s_{i-1} = d_i + s_i, \quad i \in \mathbb{I}$$

$$x_i \leq M y_i, \quad i \in \mathbb{I}$$

$$\sum_{i=1}^{T} \alpha_i \leq \beta,$$

$$x,\ w,\ s \in \mathbb{R}_+^T,\ y,\ \alpha \in \{0,1\}^T.$$

We can also relax α_i for each $i \in \mathbb{I}$ to be fractional and the following conclusion holds.

Proposition 3.1. *For a multi-stage robust lot-sizing problem with outsourcing, the formulation can be simplified as the following two-stage min-max problem:*

$$\min_{x,s,y,w} \left(\sum_{i=1}^{T} p_i x_i + h_i s_i + o_i w_i + f_i y_i + \max_{\alpha} \sum_{i=1}^{T} \alpha_i \left((o_i - p_i) x_i \right) \right)$$

$$\begin{aligned}
(\text{RPLS})\ \ s.t. \quad & x_i + w_i + s_{i-1} = d_i + s_i, && i \in \mathbb{I} \\
& x_i \leq M y_i, && i \in \mathbb{I} \\
& \alpha_i \leq 1, && i \in \mathbb{I} \\
& \sum_{i=1}^{T} \alpha_i \leq \beta, \\
& x,\ s,\ w,\ \alpha \in \mathbb{R}_+^T,\ y \in \{0,1\}^T.
\end{aligned}$$

Proof. We only need to prove that there exists an optimal solution for the above problem such that α^* is integral. We prove the claim by the contradiction method. For a given optimal solution $(x^*,\ s^*,\ y^*,\ w^*,\ \alpha^*)$, we can first observe that there exists an optimal solution in which $\alpha_i^* = 0$ if $(o_i - p_i)x_i^* \leq 0$. Now we prove that for the given optimal solution $(x^*,\ s^*,\ y^*,\ w^*)$, there exists a corresponding integral optimal solution α^* for the following subproblem:

$$\begin{aligned}
& \max_{\alpha} \ \sum_{i=1}^{T} \alpha_i \left((o_i - p_i) x_i \right) \\
(SUB)\ \ s.t. \ \ & \alpha_i \leq 1, && i \in \mathbb{I} \\
& \sum_{i=1}^{T} \alpha_i \leq \beta, \\
& \alpha \in \mathbb{R}_+^T.
\end{aligned}$$

If there exists a time period i in which $\alpha_i^* \in (0,1)$ and $\sum_{i=1}^T \alpha_i^* < \beta$, then we can increase α_i^* to be 1, which leads to a larger objective value for the above subproblem. Contradiction.

If there exists a time period i in which $\alpha_i^* \in (0,1)$ and $\sum_{i=1}^T \alpha_i^* = \beta$, then there must exist at least one more time period j in which $\alpha_j^* \in (0,1)$. If $(o_i - p_i)x_i^* = (o_j - p_j)x_j^*$, then we can increase α_i^* and decrease α_j^* such that either α_i^* or α_j^* becomes integral. Thus, we obtain a solution with a fewer number of fractional solutions. Following this same step, we can either obtain an integral solution with the same objective value or find a case in a certain step in which $(o_i - p_i)x_i^* \neq (o_j - p_j)x_j^*$. Under this scenario, without loss of generality, we can assume $(o_i - p_i)x_i^* > (o_j - p_j)x_j^*$. Then, we can obtain a larger objective value for the subproblem if we increase α_i^* and decrease α_j^* by a small value $\epsilon > 0$. Contradiction.

Therefore, the original conclusion holds. □

We can further utilize a primal-dual approach to generate a dual formulation of RPLS as follows:

Theorem 3.1. *The multi-stage robust lot-sizing problem with outsourcing can be transformed as the following single stage mixed-integer-programming problem.*

$$
\begin{aligned}
& \min_{x,s,y,w,q} \sum_{i=1}^{T} (p_i x_i + h_i s_i + o_i w_i + f_i y_i) + \beta q_0 + \sum_{i=1}^{T} q_i \\
\text{(RDLS)} \quad s.t. \quad & q_0 + q_i \geq (o_i - p_i) x_i, && i \in \mathbb{I} \\
& x_i + w_i + s_{i-1} = d_i + s_i, && i \in \mathbb{I} \\
& x_i \leq M y_i, && i \in \mathbb{I} \\
& x,\ s,\ w \in \mathbb{R}_+^T,\ q \in \mathbb{R}_+^{T+1},\ y \in \{0,1\}^T.
\end{aligned}
$$

Proof. It is easy to observe that (x^*, y^*, s^*, w^*) are bounded since problem parameters including demand are nonnegative and bounded. Then, from the formulation of SUB, we can observe that the subproblem is bounded and feasible since $\alpha = 0$ is an obvious

feasible solution. According to strong duality theorem, we can write down the dual formulation of (SUB) and show that RDLS can provide the same optimal objective value as the original problem RFLS. □

From above analysis, we can solve the multi-stage robust lot-sizing problem with outsourcing by a mixed integer problem. We can also observe that the following (ℓ, S)-type inequalities (see, e.g., Barany et al. 1984a) are valid for RDLS.

Proposition 3.2. *Corresponding to each time period $\ell \in \mathbb{I}$ and a given $S \subseteq \mathcal{L} = \{1, 2, \dots, \ell\}$, the following inequality*

$$\sum_{i \in S} x_i + \sum_{i \in \mathcal{L} \setminus S} d_{i\ell} y_i + \sum_{i \in \mathcal{L}} w_i \geq d_{1\ell},$$

where $d_{ij} = \sum_{k=i}^{j} d_k$, valid for RDLS.

Proof. If $y_i^* = 0$ for each $i \in \mathcal{L} \setminus S$, then $\sum_{i \in S} x_i + \sum_{i \in \mathcal{L} \setminus S} d_{i\ell} y_i + \sum_{i \in \mathcal{L}} w_i = \sum_{i \in S} x_i + \sum_{i \in \mathcal{L}} w_i = \sum_{i \in \mathcal{L}} x_i + \sum_{i \in \mathcal{L}} w_i \geq d_{1\ell}$. Otherwise, let $k = \text{argmin}\{i \in \mathcal{L} \setminus S : y_i^* = 1\}$. We have $\sum_{i \in S} x_i + \sum_{i \in \mathcal{L} \setminus S} d_{i\ell} y_i + \sum_{i \in \mathcal{L}} w_i \geq \sum_{i \in S} x_i + \sum_{i \in \mathcal{L}} w_i + d_{k\ell} \geq \sum_{i=1}^{k-1} x_i + \sum_{i \in \mathcal{L}} w_i + d_{k\ell} \geq d_{1\ell}$. The conclusion holds. □

3.4 The Robust Lot-sizing Problem with Backlogging: Single Disruption case

In this section, we discuss the robust lot-sizing problem which utilizes backlogging as the reparation approach and in which a single disruption happens. In the backlogging setting, the inventory and backlogging levels in the time periods which are after the disruption period are changed. We assume that every disruption in the lot-sizing problem with backlogging should be recovered after the disruption without the consideration of outsourcing. Under this assumption, we observe this problem and obtain the following observation result:

Observation 1: Based on the problem setting, when the disruption happens at period t, later periods should have the potential for recovery production. In the production horizon, the last period is a special period, because if production processes

in the last period in the first stage and then, the disruption occurs in the last period, i.e., T, no later period can recover its production. Thus, we assume that the last period does not produce in the first stage.

Note here, the purpose of the above assumption is to keep the problem self-contained. In this case, all setup decisions are made before the disruption time period and obtain the following observation result:

Observation 2: If period k is the last setup period in the first stage planning, then no production happens in this period. If production and disruption happen in last setup period, no later period can provide the recover production for its production.

Based on the general formulation for the lot-sizing problem with multiple disruptions and the above observations, we formulate the problem as a two-stage robust problem:

$$\min_{(x^1,y^1,s^1)} \sum_{i=1}^{T} p_i x_i^1 + \sum_{i=1}^{T}((f_i y_i^1 + h_i s_i^1 + b_i \ell_i^1) + \max_{t} \min_{(x^2,s^2,y^2)} [-\xi_t \phi_t(x^1, s^1, \ell^1) + \sum_{j=t+1}^{T} (\phi_j x_j^2 + h_j s_j^2 + b_j \ell_j^2)]$$

$$(PLS) \quad s.t. \quad x_i^1 + s_{i-1}^1 + \ell_i^1 = d_i + s_i^1 + \ell_{i+1}^1, \quad i \in \mathbb{I} \quad (3\text{–}4)$$

$$x_i^1 + s_{i-1}^1 + \ell_i^1 + x_i^2 + s_{i-1}^2 + \ell_i^2 = d_i + s_i^1 + \ell_{i+1}^2 + s_i^2 + \ell_{i+1}^2, \quad i \in \mathbb{I} \quad (3\text{–}5)$$

$$x_i^1 \leq M y_i^1, \quad i \in \mathbb{I} \quad (3\text{–}6)$$

$$x_i^2 \leq M y_i^2, \quad i \in \mathbb{I} \quad (3\text{–}7)$$

$$x^1,\ x^2,\ s^1,\ s^2 \in \mathbb{R}_+^T,\ y^1 \in \mathbb{B}^T. \quad (3\text{–}8)$$

with $x_T^1 = 0$.

In the objective function, if there is a disruption that happens at time period t, then the scheduled production at t cannot be processed, the corresponding production cost $p_t x_t$ is saved. Meanwhile, the corresponding inventory and backlogging costs which involve inventory and backlogging amounts from production amount x_t are saved. The total saved cost is evaluated by function $\phi_t(x^1, s^1, \ell^1)$. The constraints (3–4) and (3–5) guarantee the demand in every period can be satisfied before and after the disruption.

The constraints (3–6) and (3–7) keep that the production is according to the setup decision before and after the disruption, respectively.

In *PLS*, the exact production amounts provided by the disruption period t for demands in other time periods are hard to determine. Thus, the exact form of saving function $\phi_t(\dots)$ cannot be determined based on the information in *PLS*. Therefore, in the following section, we generate a reformulation to provide the exact mathematical formulation for *PLS*.

3.4.0.1 Reformulation

Let parameter $c_{i\ell}$ represent the total cost involved for a single item produced in time period i (except setup cost) to satisfy the demand in time period ℓ. For instance, for each $i \leq T$, we have $c_{i\ell} = p_i + \sum_{k=i}^{\ell} h_k$ if $\ell > i$, $c_{i\ell} = p_i + \sum_{k=\ell}^{i} b_k$ if $\ell < i$, and $c_{ii} = p_i$. If $i = T + 1$, then $c_{i\ell} = o_i + \sum_{k=\ell}^{T} b_k$. Problem parameters h_i and b_i represent the unit inventory and backlogging costs in time period i, respectively. We introduce variable $x_{i\ell}^k$ to build the relationship between the stage and production amounts; $x_{i\ell}^k$ is the production amount in period i to satisfy the demand in period ℓ at stage k. If $k = 1$, variables $x_{i\ell}^1$ are the scheduled production amount in period i to satisfy the demand in period ℓ. If $k = 2$, $x_{i\ell}^k$ is the extra production amount in period i to satisfy the unsatisfied demand in period ℓ. We let binary variable y_i and binary variable z_i indicate the first stage set-up decision and the second stage set-up requirement. If $z_i = 1$, period i can do recovery production after the disruption; otherwise, $z_i = 0$. Because all set-up decisions are made in the first stage, then, $z_i \leq y_i$, $2 \leq i \leq T$. The index t indicates the period when the disruption happens. With variable $x_{i\ell}^1$, it is clear that the production amount for time period i to satisfy demand in time period ℓ as inventory or backlogging is $x_{i\ell}^1$. Hence, we can reformulate the two-stage robust model *PLS* as follows:

$$
\begin{aligned}
\min_{x^1} \quad & \sum_{i=1}^{T-1}\sum_{\ell=1}^{T} c_{i\ell}x^1_{i\ell} + \sum_{i=1}^{T} f_i y_i \\
& + \max_{t} \min_{x^2} \sum_{i=t+1}^{T}\sum_{\ell=1}^{T} c_{i\ell}x^2_{i\ell} - \sum_{\ell=1}^{T} c_{t\ell}x^1_{t\ell}
\end{aligned}
$$

$$
\textit{(LS-S)} \quad s.t. \quad \sum_{i=1}^{T-1} x^1_{i\ell} = d_\ell, \qquad \ell \in \mathbb{I} \tag{3–9}
$$

$$
\sum_{i=t+1}^{T} x^2_{i\ell} = x^1_{t\ell}, \qquad \ell \in \mathbb{I} \tag{3–10}
$$

$$
x^1_{i\ell} \le d_i y_i \qquad \ell \in \mathbb{I} \tag{3–11}
$$

$$
x^2_{i\ell} \le d_i z_i \qquad \ell \in \mathbb{I} \tag{3–12}
$$

$$
z_i \le y_i \tag{3–13}
$$

$$
1 \le t \le T-1 \text{ and integer}, \tag{3–14}
$$

$$
y_i, z_i \in \{0,1\}, \; x^1_{i\ell}, x^2_{i\ell} \ge 0, \qquad \ell \in \mathbb{I}. \tag{3–15}
$$

In *LS-S*, constraints (3–9) and (3–10) guarantee that demand in each time period i can be satisfied and that the last period does not produce before the disruption due to Observation 1. Constraints (3–11) and (3–12) guarantee that production happens in the setup time period in the first and second stages. Constraint (3–13) forces that all set-up decisions are made before the disruption.

In order to solve this two-stage robust mixed-integer programming problem, we enumerate periods when the disruption happens to reformulate *LS-S* as a single stage linear programming problem. According to the problem setting, t represents the period of the disruption. The decision of $x^2_{i\ell}$ is based on the period when the first disruption happens. Thus, we let the production quantity be a function of the disruption. For instance, we let $x^2_{i\ell}(t)$ represent the extra production quantities after the disruption. We use these as decision variables in the formulation. Finally, for a given t, we use θ to represent the total extra cost after the first disruption. In this way, we enumerate

all possible scenarios for the period when the disruption happens and rewrite the formulation for the lot-sizing problem with single disruption and backlogging as follows:

$$\min_{x^1} \sum_{i=1}^{T-1} \sum_{\ell=1}^{T} c_{i\ell} x^1_{i\ell} + \theta$$

$$\text{s.t.} \quad \min_{x^2} \sum_{i=t+1}^{T} \sum_{\ell=1}^{T} c_{i\ell} x^2_{i\ell}(t) - \sum_{\ell=1}^{T} c_{t\ell} x^1_{t\ell} \leq \theta, \quad 1 \leq t \leq T-1 \tag{3–16}$$

$$\sum_{i=t+1}^{T} x^2_{i\ell}(t) = x_{t\ell}, \qquad \ell \in \mathbb{I}, \tag{3–17}$$

$$x^2_{i\ell}(t) \leq d_i z_i, \qquad 1 \leq t \leq T-1, \ell \in \mathbb{I} \tag{3–18}$$

Constraints (3–9), (3–11), (3–13), and (3–15).

We can claim that the above formulation is equivalent to the formulation without "min" operation on the left side of constraints (3–16) as shown in the following:

$$\sum_{i=t+1}^{T} \sum_{\ell=1}^{T} c_{i\ell} x^2_{i\ell}(t) - \sum_{\ell=1}^{T} c_{t\ell} x^1_{t\ell} \leq \theta, \quad 1 \leq t \leq T-1. \tag{3–19}$$

The reason lies in the following two facts:

In the optimal solution, for the formulation including (3–19), θ^1 should achieve the maximum value for the left side corresponding to a certain time period t. That is, there exists at least one tight inequality in (3–19) for θ.

In the optimal solution, for the formulation including (3–19), we have at least one tight inequality in which the left side achieves the minimum. Otherwise, θ can be decreased and we have a contradiction. Note here, we do not need to consider the "min" operation for the non-tight inequalities, since it will not affect the optimal objective value.

With inequality (3–19), the single disruption case can be solved by using the following mixed programming formulation (LS-S1)

$$\min_{x^1,\, \theta^1} \sum_{i=1}^{T-1} \sum_{\ell=1}^{T} c_{i\ell} x^1_{i\ell} + \theta^1$$

$$s.t. \quad \text{Constraints (3–9), (3–11), (3–13), (3–17), (3–18), (3–19).}$$

For the lot-sizing problem with a single disruption and backlogging case, based on the enumeration of the scenarios of t, we introduce an artificial variable θ^1 and there are $\mathcal{O}(T)$ constraints of (3–9), (3–13) and (3–19). For the disruption, for a given scenario of t, we also increase the dimension of the second stage decision variable $x^2_{i\ell}$ to be $x^2_{i\ell}(t)$. Therefore, there are $\mathcal{O}(T^2)$ constraints of (3–17) and $\mathcal{O}(T^3)$ constraints of (3–18).

To simplify the notation, we let $x^i_{q\ell}$ represent $x^2_{q\ell}(i)$. With equation (3–17), we substitute $x^1_{i\ell}$ by $\sum_{q=i+1}^{T} x^i_{q\ell}$. The detailed reformulation is as follows:

$$\min_{x,y,\theta} \sum_{i=1}^{T} f_i y_i + \sum_{i=1}^{T-1}\sum_{\ell=1}^{T} c_{i\ell} \sum_{q=i+1}^{T} x^i_{q\ell} + \theta$$

$$\text{s.t.} \quad \sum_{i=1}^{T-1}\sum_{q=i+1}^{T} x^i_{q\ell} = d_\ell, \qquad \ell \in \mathbb{I} \tag{3–20}$$

$$\sum_{q=i+1}^{T} x^i_{q\ell} \le d_\ell y_i, \qquad 1 \le i \le T-1,\ \ell \in \mathbb{I} \tag{3–21}$$

$$x^i_{q\ell} \le d_\ell z_q, \qquad 2 \le q \le T,\ \ell \in \mathbb{I} \tag{3–22}$$

$$z_q \le y_q, \qquad 2 \le q \le T \tag{3–23}$$

$$\sum_{q=i+1}^{T}\sum_{\ell=1}^{T} c_{q\ell} x^i_{q\ell} - \sum_{\ell=1}^{T} c_{i\ell}\Big(\sum_{q=i+1}^{T} x^i_{q\ell}\Big) \le \theta, \qquad 1 \le i \le T-1 \tag{3–24}$$

$$y_i, z_q \in \{0,1\},\ x^i_{q\ell} \ge 0, \qquad 1 \le i < T,\ 2 \le q \le T, \ell \in \mathbb{I}. \tag{3–25}$$

We divide d_ℓ on both side of constraints (3–20), (3–21), and let (3–24). Let $a^i_{q\ell} = x^i_{q\ell}/d_\ell$. Then, we obtain the new formulation for robust lot-sizing with a single disruption and backlogging as:

$$\min_{x,y,\theta} \sum_{i=1}^{T} f_i y_i + \sum_{i=1}^{T-1}\sum_{\ell=1}^{T} c_{i\ell} d_\ell \sum_{q=i+1}^{T} a^i_{q\ell} + \theta$$

$$(\textit{LS-SR}) \quad \text{s.t.} \sum_{i=1}^{T-1}\sum_{q=i+1}^{T} a^i_{q\ell} = 1, \qquad \ell \in \mathbb{I} \tag{3–26}$$

$$\sum_{q=i+1}^{T} a^i_{q\ell} \leq y_i, \qquad 1 \leq i \leq T-1,\ \ell \in \mathbb{I} \tag{3–27}$$

$$a^i_{q\ell} \leq z_q, \qquad 2 \leq q \leq T,\ \ell \in \mathbb{I} \tag{3–28}$$

$$z_q \leq y_q, \qquad 2 \leq q \leq T \tag{3–29}$$

$$\sum_{q=i+1}^{T} \sum_{\ell=1}^{T} c_{q\ell} d_\ell a^i_{q\ell} - \sum_{\ell=1}^{T} c_{i\ell} d_\ell \Big(\sum_{q=i+1}^{T} a^i_{q\ell} \Big) \leq \theta, \qquad 1 \leq i \leq T-1 \tag{3–30}$$

$$y_i \in \{0,1\},\ z_q \in \{0,1\},\ a^i_{q\ell} \geq 0, \qquad 1 \leq i \leq T-1,\ 2 \leq q \leq T, \ell \in \mathbb{I}. \tag{3–31}$$

Next, we study the structure of feasible region of *LS-SR*. Let

$$X_D = \{(x,y,z,\theta) \in R_+^{\frac{T \times T \times (T-1)}{2} + 2T} : (x,y,z,\theta) \text{ satisfies (3–26) to (3–31)}\}. \tag{3–32}$$

$$R_D = \{(x,y,z,\theta) \in R_+^{\frac{T \times T \times (T-1)}{2} + 2T} : (x,y,z,\theta) \text{ satisfies (3–26) to (3–30))}\}. \tag{3–33}$$

and P_D is the polyhedron of X_D. Note here, X_D records the feasible region of *LS-SR*. R_D is the relaxation of X_D.

Second, we derive the dimension of P_D and show which above inequalities are facet-defining inequalities.

Proposition 3.3. *The dimension of the polyhedron P_D is $\frac{T \times T \times (T-1)}{2} + 2T$.*

Proposition 3.4. (a) *For any $1 \leq i, \ell \leq T$, $i+1 \leq q \leq T$, $a^i_{q\ell} \geq 0$ defines a facet of P_D.*

(b) *For any $1 \leq i \leq T$, $y_i \leq 1$ defines a facet of P_D.*

3.4.0.2 Facet-defining Inequalities

In this section, we investigate facet-defining inequalities which are not in *LS-SR* for P_D.

Proposition 3.5. *The following inequalities*

$$\sum_{q=2}^{T} \left(z_q + \sum_{i=1}^{q-1} a^i_{q\ell(q)} \right) \geq 2, \ \textit{where } \ell(q) \in \mathbb{I} \textit{ and } \ell(q_1) \neq \ell(q_2) \textit{ if } q_1 \neq q_2 \tag{3–34}$$

are valid and facet-defining for LS-SR.

Proof. We prove Proposition 3.5 by two claims.

Claim 1. (3–34) is a valid inequality for *LS-SR*.

Claim 2. (3–34) is a facet-defining inequality for *LS-SR*.

Proof of Claim 1. From Observations 1 and 2, we know that in order to recover the production, at least one period is set up only for the recovery production. Thus $\sum_{q=2}^{T} z_q \geq 1$. We discuss the following two cases.

Case 1. If there is more than one period which is set up for recovery production, then $\sum_{q=2}^{T} z_q \geq 2$.

Case 2. If there is only one setup period for recovery production, then $\sum_{i=2}^{T} z_i = 1$. With (3–29), we have $\sum_{q=2}^{T} \sum_{i=1}^{q-1} a^i_{q\ell(q)} = 1$. Because if $\sum_{q=2}^{T} \sum_{i=1}^{q-1} a^i_{q\ell(q)} \neq 1$, demand in each period cannot be satisfied.

Therefore, (3.5) is a valid inequality for *LS-SR*.

Proof of Claim 2. Let $\rho z + \beta a + \gamma y + \kappa\theta \leq \delta$ be a valid inequality for P_D and assume that

$$\mathcal{R}_1 = \{(x, y, z, \theta) \in P_D, \sum_{q=2}^{T}(z_q + \sum_{i=1}^{q-1} a^i_{q\ell(q)}) = 2\} \tag{3–35}$$

$$\subseteq \{(x, y, z, \theta) \in P_D, \sum_{i=2}^{T} \rho_i z_i + \sum_{i=1}^{T-1} \sum_{q=i+1}^{T} \sum_{\ell=1}^{T} \beta^i_{q\ell} a^i_{q\ell} + \sum_{i=1}^{T} \gamma_i y_i + \kappa\theta = \eta\} \tag{3–36}$$

We prove that (3–34) represents a facet for P_D by showing

$$\kappa = 0, \tag{3–37}$$

$$\gamma_i = 0, \qquad i = 1, \cdots, T \tag{3–38}$$

$$\rho_i = -\rho, \qquad i = 2, \cdots, T \tag{3–39}$$

$$\beta^i_{q\ell} = -\rho + u_\ell, \qquad \ell = \ell(q) \tag{3–40}$$

$$\beta^i_{q\ell} = u_\ell, \qquad \ell \neq \ell(q) \tag{3–41}$$

$$\eta = -2\rho + \sum_{l=1}^{T} u_\ell. \tag{3–42}$$

First, we prove that $\kappa = 0$. With a feasible and tight solution, (x, y, z, M) for $\mathcal{R}_1$, we have another feasible and tight solution $(z, x, y, 2M)$ for $\mathcal{R}_1$, where $M = \max_{(i,\ell)} c_{i\ell} \sum_{i=1}^{T} d_i$. Then, we have

$$\sum_{i=2}^{T} \rho_i z_i + \sum_{i=1}^{T-1} \sum_{q=i+1}^{T} \sum_{\ell=1}^{T} \beta_{q\ell}^i a_{q\ell}^i + \sum_{i=1}^{T} \gamma_i y_i + \kappa M = \eta$$
$$\sum_{i=2}^{T} \rho_i z_i + \sum_{i=1}^{T-1} \sum_{q=i+1}^{T} \sum_{\ell=1}^{T} \beta_{q\ell}^i a_{q\ell}^i + \sum_{i=1}^{T} \gamma_i y_i + 2\kappa M = \eta$$

Thus, $\kappa = 0$. We assume $\theta = M$ in the following discussion.

Second, we prove that $\gamma_i = 0$ for all $i = 1, \cdots, T$. We construct two tight feasible solutions for $\mathcal{R}_1$ as following:

Point 1: $y_1 = y_2 = \cdots = y_T = 1$, $z_{q_1} = z_{q_2} = 1$, $a_{q_1 k}^i = 1$, $a_{q_2 \ell(q_1)}^i = 1$, $k \neq \ell(q_1)$, $q_1, q_2 \leq T - 1$.

Point 2: $y_1 = y_2 = \cdots = y_{T-1} = 1$, $z_{q_1} = z_{q_2} = 1$, $a_{q_1 k}^i = 1$, $a_{q_2 \ell(q_1)}^i = 1$, $k \neq \ell(q_1)$, $q_1, q_2 \leq T - 1$.

Then, we have

$$\sum_{i=1}^{T} \rho_i z_i + \sum_{i=1}^{T-1} \sum_{q=i+1}^{T} \sum_{\ell=1}^{T} \beta_{q\ell}^i a_{q\ell}^i + \sum_{i=1}^{T} \gamma_i = \eta$$
$$\sum_{i=1}^{T} \rho_i z_i + \sum_{i=1}^{T-1} \sum_{q=i+1}^{T} \sum_{\ell=1}^{T} \beta_{q\ell}^i a_{q\ell}^i + \sum_{i=1}^{T-1} \gamma_i = \eta$$

Thus, $\gamma_T = 0$. With different construction of tight feasible solutions, we have $\gamma_i = 0$. We assume $y = 1$ in the following discussion.

Now, we show the following coefficient relationship among β, ρ and η. We construct the following tight feasible solutions for $\mathcal{R}_1$:

Point 3: $y_{i_1} = y_{i_2} = y_{j_1} = y_{j_2} = 1$, $z_{j_1} = z_{j_2} = 1$, $a_{j_1 \ell(j_2)}^{i_1} = 1$, $a_{j_2 \ell(j_1)}^{i_1} = 1$, $a_{j_2 k}^{i_1} = 1$, $a_{j_2 \ell}^{i_1} = 1$, where $i_1, i_2 \leq j_1, j_2$, $k \neq \ell(j_1), \ell(j_2)$, $\ell \neq \ell(j_1), \ell(j_2), k$.

Point 4: $y_{i_1} = y_{i_2} = y_{j_1} = y_{j_2} = 1$, $z_{j_1} = z_{j_2} = 1$, $a_{j_1 \ell(j_2)}^{i_1} = 1$, $a_{j_2 \ell(j_1)}^{i_1} = 1$, $a_{j_2 k}^{i_2} = 1$, $a_{j_2 \ell}^{i_1} = 1$, where $i_1, i_2 \leq j_1, j_2$, $k \neq \ell(j_1), \ell(j_2)$, $\ell \neq \ell(j_1), \ell(j_2), k$.

Point 5: $y_{i_1} = y_{i_2} = y_{j_1} = y_{j_2} = 1$, $z_{j_1} = z_{j_2} = 1$, $a^{i_1}_{j_1\ell(j_2)} = 1$, $a^{i_1}_{j_2\ell(j_1)} = 1$, $a^{i_1}_{j_1\ell} = 1$, where $i_1, i_2 \leq j_1, j_2$, $\ell \neq \ell(j_1), \ell(j_2)$.

Point 6: $y_{i_1} = y_{i_2} = y_{j_1} = y_{j_3} = 1$, $z_{j_1} = z_{j_3} = 1$, $a^{i_1}_{j_1\ell(j_3)} = 1$, $a^{i_1}_{j_3\ell(j_1)} = 1$, $a^{i_1}_{j_3k} = 1$, $a^{i_1}_{j_3\ell} = 1$, where $i_1, i_2 \leq j_1, j_3$, $\ell \neq \ell(j_1), \ell(j_2)$.

Point 7: $y_{i_1} = y_{i_2} = y_{j_1} = y_{j_2} = 1$, $z_{j_1} = 1$, $a^{i_1}_{j_1\ell} = 1$, where $i_1, i_2 \leq j_1, j_2$.

Point 8: $y_{i_1} = y_{i_2} = y_{j_1} = y_{j_2} = 1$, $z_{j_1} = z_{j_2} = 1$, $a^{i_1}_{j_1\ell(j_2)} = 1 a^{i_2}_{j_2k} = 1$, $a^{i_1}_{j_2\ell} = 1$, where $i_1, i_2 \leq j_1, j_2$, $k \neq \ell(j_2)$, $\ell \neq \ell(j_2), k$.

We show that $\beta^{i_1}_{q\ell} = \beta^{i_2}_{q\ell} = \beta_{q\ell}$ for $\ell \neq \ell(q)$ by putting Point 3, Point 4, and Point 8 into

$$\sum_{i=2}^{T} \rho_i z_i + \sum_{i=1}^{T-1} \sum_{q=i+1}^{T} \sum_{\ell=1}^{T} \beta^{i}_{q\ell} a^{i}_{q\ell} = \eta. \tag{3–43}$$

Then, we have

$$\rho_{j_1} + \rho_{j_2} + \beta^{i_1}_{j_1\ell(j_2)} + \beta^{i_1}_{j_2\ell(j_1)} + \beta^{i_1}_{j_2k} + \sum_{\ell\neq\ell(j_1),\ell(j_2),k} \beta^{i_1}_{j_1\ell} = \eta, \tag{3–44}$$

$$\rho_{j_1} + \rho_{j_2} + \beta^{i_1}_{j_1\ell(j_2)} + \beta^{i_1}_{j_2\ell(j_1)} + \beta^{i_2}_{j_2k} + \sum_{\ell\neq\ell(j_1),\ell(j_2),k} \beta^{i_1}_{j_1\ell} = \eta, \tag{3–45}$$

$$\rho_{j_1} + \rho_{j_2} + \beta^{i_2}_{j_2k} + \beta^{i_1}_{j_1\ell(j_2)} + \sum_{\ell\neq\ell(j_2),k} \beta^{i_1}_{j_2\ell} = \eta. \tag{3–46}$$

Thus, with (3–44) and (3–45), we have $\beta^{i_1}_{j_2k} = \beta^{i_2}_{j_2k} = \beta_{j_2k}$, where $k \neq \ell(j_1), \ell(j_2)$. With (3–44) and (3–46), we have $\beta^{i_2}_{j_2k} = \beta^{i_1}_{j_2k}$, where $k \neq \ell(j_1)$. By the arbitrary construction of Point 3, 4, and 8, we have

$$\beta^{i_1}_{q\ell} = \beta^{i_2}_{q\ell} = \beta_{q\ell}, \text{ for } \ell \neq \ell(q). \tag{3–47}$$

We show that $\beta^{i}_{j_1\ell} = \beta^{i}_{j_2\ell} = \beta^{i}_{\ell}$. we put points 3 and 5 into (3–43), we have (3–44) and

$$\rho_{j_1} + \rho_{j_2} + \beta^{i_1}_{j_1\ell(j_2)} + \beta^{i_1}_{j_2\ell(j_1)} + \beta^{i_1}_{j_1k} + \sum_{\ell\neq\ell(j_1),\ell(j_2),k} \beta^{i_1}_{j_1\ell} = \eta. \tag{3–48}$$

With different constructions of feasible solutions, we have

$$\beta^{i_1}_{j_1k} = \beta^{i_1}_{j_2k} = \beta^{i_1}_{k}, \text{ where } k \neq \ell(j_1), \ell(j_2). \tag{3–49}$$

With (3–47) and (3–49), we have $\beta_{j_1 k} = \beta^{i_1}_{j_1 k} = \beta^{i_1}_{j_2 k} = \beta^{i_1}_{k}$. Thus, the following conclusion holds.

$$\beta^{i}_{j\ell} = u_\ell, \qquad \ell \neq \ell(j). \tag{3–50}$$

Put Point 5 and Point 6 into (3–43), with (3–50), we have $\rho_{j_2} = \rho_{j_3}$.With the different construction, we have

$$\rho_j = -\rho.$$

Put points 3 and 7 into (3–43), we have $\beta^{i}_{j\ell} = -\rho + u_\ell$, where $\ell = \ell(j)$. Hence,

$$\beta^{i}_{j\ell(j)} = -\rho + u_\ell.$$

With any tight feasible solution for $\mathcal{R}_1$, we have $\eta = -2\rho + \sum_{\ell=1}^{T} u_\ell$. □

Based on Proposition 3.5 and constraint (3–23), we generate the following facet-defining inequality for P_D as follows.

Proposition 3.6. *The following inequalities*

$$y_1 + \sum_{q=2}^{T}\left(y_q + \sum_{i=1}^{q-1} a^{i}_{q\ell(q)}\right) \geq 3, \text{ for all } \ell(q) \in \mathbb{I} \text{ such that } \ell(q_1) \neq \ell(q_2) \text{ if } q_1 \neq q_2 \tag{3–51}$$

are valid and facet-defining for LS-SR.

Proof. We proof Proposition 3.6 by two claims.

Claim 1. (3–51) is a valid inequality for *LS-SR*.

Claim 2. (3–51) is a facet-defining inequality for *LS-SR*.

Proof of Claim 1. From Observations 1 and 2, in order to satisfy demands and do recovery production, $\sum_{i=1}^{T} y_i \geq 2$. We prove Claim 1 by two cases.

Case 1. If there are more than two production periods, then $\sum_{i=1}^{T} y_i \geq 3$.

Case 2. If there are only two setup periods, the recovery period covers the demand in period $\ell(q)$ to satisfy the constraint (3–20). Thus, (3–49) holds.

Therefore, (3–51) is a valid inequality.

Proof of Claim 2. Let $\rho z + \beta a + \gamma y + \kappa\theta \leq \delta$ be a valid inequality for P_D and assume that

$$\mathcal{R}_2 = \{(x, y, z, \theta) \in P_D, y_1 + \sum_{q=2}^{T}(y_q + \sum_{i=1}^{q-1} a^i_{q\ell(q)}) = 3\} \tag{3–52}$$

$$\subseteq \{(x, y, z, \theta) \in P_D, \sum_{i=2}^{T} \rho_i z_i + \sum_{q=2}^{T}\sum_{i=1}^{q-1}\sum_{\ell=1}^{T} \beta^i_{q\ell} a^i_{q\ell} + \sum_{i=1}^{T} \gamma_i y_i + \kappa\theta = \eta\} \tag{3–53}$$

We prove that (3–51) represents a facet by showing that

$$\kappa = 0, \tag{3–54}$$

$$\rho_i = 0, \qquad i = 2, \cdots, T \tag{3–55}$$

$$\gamma_i = -\alpha, \qquad i = 1, \cdots, T \tag{3–56}$$

$$\beta^i_{q\ell} = -\alpha + u_\ell, \qquad \ell = \ell(q) \tag{3–57}$$

$$\beta^i_{q\ell} = u_\ell, \qquad \ell \neq \ell(q) \tag{3–58}$$

$$\eta = -3\alpha + \sum_{\ell=1}^{T} u_\ell \tag{3–59}$$

First, we prove that $\kappa = 0$. With a feasible and tight solution (x, y, z, M) for $\mathcal{R}_2$, we have another feasible and tight solution $(x, y, z, 2M)$ for $\mathcal{R}_2$, where $M = \max_{(i,\ell)} c_{i\ell} \sum_{i=1}^{T} d_i$. Then, we have

$$\sum_{i=2}^{T} \rho_i z_i + \sum_{i=1}^{T-1}\sum_{q=i+1}^{T}\sum_{\ell=1}^{T} \beta^i_{q\ell} a^i_{q\ell} + \sum_{i=1}^{T} \gamma_i y_i + \kappa M = \eta$$
$$\sum_{i=2}^{T} \rho_i z_i + \sum_{i=1}^{T-1}\sum_{q=i+1}^{T}\sum_{\ell=1}^{T} \beta^i_{q\ell} a^i_{q\ell} + \sum_{i=1}^{T} \gamma_i y_i + 2\kappa M = \eta.$$

Thus, $\kappa = 0$. We assume $\theta = M$ in the following discussion.

Second, we show that $\rho_i = 0$ for $2 \leq i \leq T$. We construct the following two feasible points of $\mathcal{R}_2$.

Point 1: $y_{i_1} = y_{i_2} = 1$, $z_{i_2} = 1$, $a^{i_1}_{i_2\ell} = 1$, $1 \leq i_1 < i_2 \leq T$.

Point 2: $y_{i_1} = y_{i_2} = 1$, $z_{i_1} = z_{i_2} = 1$, $a^{i_1}_{i_2\ell} = 1$, $1 \leq i_1 < i_2 \leq T$.

Put points 1 and 2 into

$$\sum_{i=2}^{T} \rho_i z_i + \sum_{i=1}^{T-1} \sum_{q=i+1}^{T} \sum_{\ell=1}^{T} \beta_{q\ell}^{i} a_{q\ell}^{i} + \sum_{i=1}^{T} \gamma_i y_i = \eta,$$

we have $\rho_{i_1} = 0$. For the arbitrary construction of points 1 and 2, we have $\rho_i = 0$.

Now, we show the following relationship among β, γ and η. We construct following tight feasible solutions for $\mathcal{R}_2$.

Point 3: $y_{i_1} = 1$, $y_{j_1} = y_{j_2} = 1$, $x_{j_1\ell(j_2)}^{i_1} = 1$, $x_{j_2\ell(j_1)}^{i_1} = 1$, $x_{j_2k}^{i_1} = 1$, $x_{j_1\ell}^{i_1} = 1$, where $j_1, j_2 > 1$, $k \neq \ell(j_1), \ell(j_2)$, $\ell \neq \ell(j_1), \ell(j_2), k$.

Point 4: $y_{i_1} = 1$, $y_{j_1} = y_{j_2} = 1$, $x_{j_1\ell(j_2)}^{i_1} = 1$, $x_{j_2\ell(j_1)}^{i_1} = 1$, $x_{j_2k}^{j_1} = 1$, $x_{j_1\ell}^{i_1} = 1$, where $j_1, j_2 > 1$, $k \neq \ell(j_1), \ell(j_2)$, $\ell \neq \ell(j_1), \ell(j_2), k$.

Point 5: $y_{i_1} = 1$, $y_{j_1} = y_{j_2} = 1$, $x_{j_1\ell(j_2)}^{i_1} = 1$, $x_{j_2\ell(j_1)}^{i_1} = 1$, $x_{j_1k}^{i_1} = 1$, $x_{j_1\ell}^{i_1} = 1$, where $j_1, j_2 > 1$, $k \neq \ell(j_1), \ell(j_2)$, $\ell \neq \ell(j_1), \ell(j_2), k$.

Point 6: $y_{i_1} = 1$, $y_{j_1} = y_{j_3} = 1$, $x_{j_1\ell(j_3)}^{i_1} = 1$, $x_{j_3\ell(j_1)}^{i_1} = 1$, $x_{j_1\ell}^{i_1} = 1$, where $j_1, j_3 > 1$, $\ell \neq \ell(j_1), \ell(j_3)$.

Point 7: $y_{i_1} = y_{j_1} = 1$, $x_{j_1\ell}^{i_1} = 1$.

First, we show that $\beta_{jk}^{i_1} = \beta_{jk}^{i_2} = \beta_{jk}$, $k \neq \ell(j)$. Put points 3 and 4 into

$$\sum_{i=1}^{T} \gamma_i y_i + \sum_{i=1}^{T-1} \sum_{q=i+1}^{T} \sum_{\ell=1}^{T} \beta_{q\ell}^{i} a_{q\ell}^{i} = \eta. \tag{3–60}$$

Then, we have

$$\gamma_{i_1} + \gamma_{j_1} + \gamma_{j_2} + \sum_{\ell \neq \ell(j_1), \ell(j_2), k} \beta_{j_1\ell}^{i_1} + \beta_{j_2\ell(j_1)}^{i_1} + \beta_{j_1\ell(j_2)}^{i_1} + \beta_{j_2k}^{i_1} = \eta$$

$$\gamma_{i_1} + \gamma_{j_1} + \gamma_{j_2} + \sum_{\ell \neq \ell(j_1), \ell(j_2), k} \beta_{j_1\ell}^{i_1} + \beta_{j_2\ell(j_1)}^{i_1} + \beta_{j_1\ell(j_2)}^{j_1} + \beta_{j_2k}^{j_1} = \eta$$

We have $\beta_{j_2k}^{i_1} = \beta_{j_2k}^{j_1} = \beta_{j_2k}$, where $k \neq \ell(j_2)$. Due to the arbitrary construction of points 3 and 4, then we have $\beta_{jk}^{i_1} = \beta_{jk}^{i_2} = \beta_{jk}$, where $j > 2$, $k \neq \ell(j)$. Note here, under the problem setting, $\beta_{2\cdot}^{i_1}$ is a $T \times 1$ vector.

Second, we show that $\beta_{j_1 k} = \beta_{j_2 k} = u_k$, where $k \neq \ell(j_1), \ell(j_2)$. Put points 3 and Point 5 into (3–60). Then,

$$\gamma_{i_1} + \gamma_{j_1} + \gamma_{j_2} + \sum_{\ell \neq \ell(j_1), \ell(j_2), k} \beta^{i_1}_{j_1 \ell} + \beta^{i_1}_{j_2 \ell(j_1)} + \beta^{i_1}_{j_1 \ell(j_2)} + \beta^{i_1}_{j_2 k} = \eta$$

$$\gamma_{i_1} + \gamma_{j_1} + \gamma_{j_2} + \sum_{\ell \neq \ell(j_1), \ell(j_2), k} \beta^{i_1}_{j_1 \ell} + \beta^{i_1}_{j_2 \ell(j_1)} + \beta^{i_1}_{j_1 \ell(j_2)} + \beta^{i_1}_{j_1 k} = \eta$$

We have $\beta^{i_1}_{j_2 k} = \beta^{i_1}_{j_1 k}$. Due to $\beta^{i_1}_{j_2 k} = \beta_{j_2 k}$, and $\beta^{i_1}_{j_1 k} = \beta_{j_1 k}$. Thus, $\beta_{j_2 k} = \beta_{j_1 k} = u_k$, where $k \neq \ell(j_1)$, or $\ell(j_2)$. Due to the arbitrary choice of j_1 and j_2, we have $\beta_{jk} = u_k$, where $k \neq \ell(j)$.

Third, we show that $\gamma_i = \gamma_j = -\alpha$. We put points 5 and 6 into (3–60). Then,

$$\gamma_{i_1} + \gamma_{j_1} + \gamma_{j_2} + \sum_{\ell \neq \ell(j_1), \ell(j_2)} \beta^{i_1}_{j_1 \ell} + \beta^{i_1}_{j_2 \ell(j_1)} + \beta^{i_1}_{j_1 \ell(j_2)} = \eta$$

$$\gamma_{i_1} + \gamma_{j_1} + \gamma_{j_3} + \sum_{\ell \neq \ell(j_1), \ell(j_3)} \beta^{i_1}_{j_1 \ell} + \beta^{i_1}_{j_3 \ell(j_1)} + \beta^{i_1}_{j_1 \ell(j_3)} = \eta$$

Because $\beta^{i}_{j\ell} = u_\ell$, where $\ell \neq \ell(j_1)$, With $\beta^{i_1}_{j_2 \ell(j_1)} = \beta^{i_1}_{j_3 \ell(j_1)} = u_{\ell(j_1)}$, $\gamma_{j_2} = \gamma_{j_3}$. With the arbitrary choice of j_2 and j_3, we have $\gamma_j = -\alpha$.

Forth, we show that $\beta^{i}_{j\ell(j)} = -\alpha + u_{\ell(j)}$. Put points 3 and 7 to (3–60). We have

$$\gamma_{i_1} + \gamma_{j_1} + \gamma_{j_2} + \sum_{\ell \neq \ell(j_1), \ell(j_2)} \beta^{i_1}_{j_1 \ell} + \beta^{i_1}_{j_2 \ell(j_1)} + \beta^{i_1}_{j_1 \ell(j_2)} = \eta$$

$$\gamma_{i_1} + \gamma_{j_1} + \sum_{\ell \neq \ell(j_1), \ell(j_2)} \beta^{i_1}_{j_1 \ell} + \beta^{i_1}_{j_1 \ell(j_1)} + \beta^{i_1}_{j_1 \ell(j_2)} = \eta$$

Then, $\beta^{i_1}_{j_1 \ell(j_1)} = -\gamma_{j_2} + \beta^{i_1}_{j_2 \ell(j_1)} = -\alpha + u_{\ell(j_1)}$. With the arbitrary choice of j_1 and j_2, we have $\beta^{i}_{j\ell(j)} = -\alpha + u_{\ell(j)}$.

Finally, put any tight feasible solutions. Then, we get $\eta = -3\alpha + \sum_{\ell=1}^{T} u_\ell$.

Therefore, (3–51) is a facet-defining inequality. □

The above two facet-defining inequalities are generated based on the relationship between the recovery production and recovery setup requirement. Now, we construct

a facet-defining inequality for P_D based on the relationship between the production and the setup decision.

Proposition 3.7. *The following inequalities*

$$\sum_{k=q+1}^{T} a_{k\ell}^{q} + \sum_{i=1}^{q-1} a_{q\ell}^{i} \leq y_q, \ 2 \leq q \leq T, \ 1 \leq \ell \leq T, \tag{3–61}$$

are valid and facet-defining for LS-SR*.*

Proof. We prove that Proposition 3.7 holds by two claims.

Claim 1. (3–61) is a valid inequality for *LS-SR*.

Claim 2. (3–61) is a facet-defining inequality for *LS-SR*.

Proof of Claim 1. If $y_q = 0$, we have $\sum_{k=q+1}^{T} a_{k\ell}^{q} = 0$ and $z_q = 0$. Thus, $\sum_{k=q+1}^{T} a_{k\ell}^{q} + \sum_{i=q+1}^{T} a_{q\ell}^{i} = 0$. If $y_q = 1$, because

$$a_{q\ell}^{i} \leq \sum_{q=i+1}^{T} a_{q\ell}^{i} \text{ and } \sum_{i=1}^{T-1} \sum_{q=i+1}^{T} a_{q\ell}^{i} = 1.$$

Then, we have $\sum_{k=q+1}^{T} a_{k\ell}^{q} + \sum_{i=q+1}^{T} a_{q\ell}^{i} \leq 1$. Therefore, (3–61) is a valid inequality for *LS-SR*.

Proof of Claim 2. Let $\rho z + \beta a + \gamma y + \kappa\theta \leq \delta$ be a valid inequality for P_D and assume that

$$\begin{aligned}\mathcal{R}_3 &= \{(x, y, z, \theta) \in P_D, \sum_{k=q+1}^{T} a_{k\ell}^{q} + \sum_{i=1}^{q-1} a_{q\ell}^{i} = y_q\} \\ &\subseteq \{(x, y, z, \theta) \in X_D, \sum_{i=2}^{T} \rho_i z_i + \sum_{q=2}^{T} \sum_{i=1}^{q-1} \sum_{\ell=1}^{T} \beta_{q\ell}^{i} a_{q\ell}^{i} + \sum_{i=1}^{T} \gamma_i y_i + \kappa\theta = \delta\}\end{aligned}$$

We prove that (3–61) with a pair of (ℓ, q) represents a facet-defining inequality by showing

$$\kappa = 0,$$

$$\eta = \sum_{\ell=1}^{T} \varphi_\ell$$

$$
\begin{aligned}
&\gamma_q = \gamma_q && \\
&\rho_i = 0, && i = 2, \cdots, T \\
&\gamma_i = 0, && i \neq q \\
&\beta^i_{q\ell} = -\gamma_q + \varphi_\ell, && i = 1, \cdots, q-1 \\
&\beta^q_{k\ell} = -\gamma_q + \varphi_\ell, && k = q+1, \cdots, T \\
&\beta^i_{jh} = \varphi_h, && j \neq q, \text{ or } i \neq q.
\end{aligned}
$$

We prove that $\kappa = 0$. With a feasible and tight solution, (x, y, z, M) for $\mathcal{R}_3$, we have another feasible and tight solution $(x, y, z, 2M)$ for $\mathcal{R}_3$, where $M = \max_{(i,\ell)} c_{i\ell} \sum_{i=1}^{T} d_i$. Then, we have

$$
\begin{aligned}
\sum_{i=2}^{T} \rho_i z_i + \sum_{i=1}^{T-1} \sum_{q=i+1}^{T} \sum_{\ell=1}^{T} \beta^i_{q\ell} a^i_{q\ell} + \sum_{i=1}^{T} \gamma_i y_i + \kappa M = \eta \\
\sum_{i=2}^{T} \rho_i z_i + \sum_{i=1}^{T-1} \sum_{q=i+1}^{T} \sum_{\ell=1}^{T} \beta^i_{q\ell} a^i_{q\ell} + \sum_{i=1}^{T} \gamma_i y_i + 2\kappa M = \eta
\end{aligned}
$$

Thus, $\kappa = 0$. We assume $\theta = M$ in the following discussion.

We show that $\rho_i = 0$ for $2 \leq i \leq T$. We construct the following two feasible points of $\mathcal{R}$:

Point 1: $y_{i_1} = y_{j_1} = y_{j_2} = 1$, $z_{j_2} = 1$, $a^{j_1}_{j_2 k} = 1$, $1 \leq i_1 < j_1, j_2 \leq T$, $1 \leq k \leq T$.

Point 2: $y_{i_1} = y_{j_1} = y_{j_2} = 1$, $z_{j_1} = z_{j_2} = 1$, $a^{j_1}_{j_2 k} = 1$, $1 \leq i_1 < j_1, j_2 \leq T$, $1 \leq k \leq T$.

Put points 1 and 2 into

$$
\sum_{i=2}^{T} \rho_i z_i + \sum_{i=1}^{T-1} \sum_{q=i+1}^{T} \sum_{\ell=1}^{T} \beta^i_{q\ell} a^i_{q\ell} + \sum_{i=1}^{T} \gamma_i y_i = \eta,
$$

we have $\rho_{j_1} = 0$. For the arbitrary construction of points 1 and 2, we have $\rho_i = 0$.

We show the relationship of β, γ, and η with equation

$$
\sum_{i=1}^{T-1} \sum_{q=i+1}^{T} \sum_{\ell=1}^{T} \beta^i_{q\ell} a^i_{q\ell} + \sum_{i=1}^{T} \gamma_i y_i = \eta. \tag{3–62}
$$

We construct the following feasible solutions for $\mathcal{R}_3$:

Point 3: $y_{i_1} = y_q = 1$, $a^{i_1}_{qk} = 1$, $1 \le k \le T$.

Point 4: $y_{i_1} = y_q = y_{j_1} = 1$, $a^{i_1}_{qk} = 1$, $1 \le k \le T$.

Point 5: $y_{i_1} = y_{i_2} = y_q = 1$, $a^{i_2}_{q\ell} = 1$, $a^{i_2}_{qk} = 1$, where $k \neq \ell$.

Point 6: $y_{i_1} = y_{i_2} = y_q = 1$, $a^{i_1}_{q\ell} = 1$, $a^{i_2}_{qk} = 1$, where $k \neq \ell$.

Point 7: $y_{i_1} = y_q = y_{j_1} = 1$, $a^{q}_{j_1\ell} = 1$, $a^{i_2}_{qk} = 1$, where $k \neq \ell$.

Point 8: $y_{i_1} = y_{j_1} = y_{j_2} = 1$, $a^{i_1}_{j_1 k} = 1$, $1 \le k \le T$, $i_1, j_1, j_2 \neq q$.

Point 9: $y_{i_1} = y_q = y_{j_1} = 1$, $a^{i_1}_{q\ell} = 1$, $a^{i_1}_{j_1 k} = 1$, $k \neq \ell$.

Point 10: $y_{i_1} = y_{j_1} = y_{j_2} = 1$, $a^{i_1}_{j_2 m} = 1$, $a^{i_1}_{j_1 k} = 1$, $i_1, j_1, j_2 \neq q$, $k \neq m$.

Point 11: $y_{i_1} = y_{j_1} = y_{j_2} = 1$, $a^{i_1}_{j_2 k} = 1$, $1 \le k \le T$, $i_1, j_1, j_2 \neq q$.

Point 12: $y_{i_1} = y_{j_1} = y_{j_2} = 1$, $a^{j_1}_{j_2 m} = 1$, $a^{i_1}_{j_2 k} = 1$, $i_1, j_1, j_2 \neq q$, $k \neq m$.

We show that $\gamma_i = 0$, where $i \neq q$. Put points 3 and 4 into (3–62). We get

$$\gamma_{i_1} + \gamma_q + \sum_{\ell=1}^{T} \beta^{i_1}_{q\ell} = \eta,$$

$$\gamma_{i_1} + \gamma_q + \gamma_{j_1} + \sum_{\ell=1}^{T} \beta^{i_1}_{q\ell} = \eta.$$

Then, we have $\gamma_{j_1} = 0$, $j_1 \neq q$ With the arbitrary construction of points 3 and 4, we have $\gamma_i = 0$, where $i \neq q$.

We show that $\beta^{i_1}_{q\ell} = \beta^{i_2}_{q\ell} = \beta_{q\ell}$. With $y_i = 0$, $i \neq q$, we put points 4 and 5 into (3–62) and obtain

$$\gamma_q + \beta^{i_1}_{q\ell} + \sum_{k \neq \ell} \beta^{i_1}_{q\ell} = \eta$$

$$\gamma_q + \beta^{i_2}_{q\ell} + \sum_{k \neq \ell} \beta^{i_2}_{q\ell} = \eta$$

Then, we have $\beta^{i_1}_{q\ell} = \beta^{i_2}_{q\ell} = \beta_{q\ell}$.

We show that $\beta^{q}_{k\ell} = \beta^{i}_{q\ell}$, where $i \le q \le k$. We put points 6 and 7 into (3–62) and obtain

$$\gamma_q + \beta^{i_1}_{q\ell} + \sum_{k \neq \ell} \beta^{i_2}_{qk} = \eta$$

$$\gamma_q + \beta^{q}_{j_1\ell} + \sum_{k \neq \ell} \beta^{i_2}_{qk} = \eta$$

Then, we have $\beta^{i_1}_{q\ell} = \beta^{q}_{j_1\ell} = \beta_{q\ell}$.

We show that $\beta^{i}_{j\ell} = \varphi_\ell$, where $i, j \neq q$. Put points 8 and 10 into (3–61), we have $\beta^{i_1}_{j_2 k} = \beta^{i_1}_{j_1 k} = \beta^{i_1}_{k}$. With the different construction of points 8 and 10, we have $\beta^{i}_{jk} = \beta^{i}_{k}$, where $i, j \neq q$.

Put points 11 and 12 into (3–61), we have $\beta^{i_1}_{k} = \beta^{j_1}_{k} = \varphi_k$, where $i_1, j_1 \neq q$. With the different construction of points, we have $\beta^{i}_{jk} = \varphi_k$, where $i, j \neq q$.

We show that $\beta_{q\ell} = \phi_\ell - \gamma q$. Put points 8 and 9 into (3–61) and obtain

$$\gamma_q + \beta^{i_1}_{q\ell} + \sum_{k \neq \ell} \beta^{i_1}_{j_1 k} = \eta$$

$$\sum_{k=1}^{T} \beta^{i_1}_{j_1 k} = \eta$$

Then, we have $\gamma_q + \beta^{i_1}_{q\ell} = \beta^{i_1}_{j_1\ell} = \gamma_q + \beta_{q\ell} = \varphi_\ell$. Thus, $\beta_{q\ell} = \varphi_\ell - \gamma_q$.

Finally, we put any feasible solution of $\mathcal{R}_3$ into (3–61) and obtain $\eta = \sum_{\ell=1}^{T} \varphi_\ell$. □

3.5 The Robust Lot-sizing Problem with Backlogging: Multiple Disruption case

In this section, we present the formulation for the robust lot-sizing problem with multiple disruptions, and with and without setup cost cases, respectively.

3.5.1 Without Setup Cost Case

Let the index t_i indicate the period when the ith disruption happens, where $i = 1, \cdots, \beta$. We extend the three-stage robust model PLS to multi-stage, and formulate the robust lot-sizing problem with multiple disruptions and without setup cost case as follows:

$$
\begin{aligned}
\min_{x^1} \; & \sum_{i=1}^{T-1} \sum_{\ell=1}^{T} c_{i\ell} x_{i\ell}^1 \\
& + \max_{t_1} \min_{x^2} \sum_{i=t_1+1}^{T-1} \sum_{\ell=1}^{T} c_{i\ell} x_{i\ell}^2 - \sum_{\ell=1}^{T} c_{t_1\ell} x_{t_1\ell}^1 \\
& + \max_{t_2>t_1} \min_{x^3} \sum_{i=t_2+1}^{T-1} \sum_{\ell=1}^{T} c_{i\ell} x_{i\ell}^3 - \sum_{\ell=1}^{T} c_{t_2\ell} \left(\sum_{k=1}^{2} x_{t_2\ell}^k \right) \\
& + \cdots\cdots \\
& + \max_{t_\beta>t_{\beta-1}} \min_{x^{\beta+1}} \sum_{i=t_\beta+1}^{T} \sum_{\ell=1}^{T} c_{i\ell} x_{i\ell}^{\beta+1} - \sum_{\ell=1}^{T} c_{t_\beta\ell} \left(\sum_{k=1}^{\beta} x_{t_\beta\ell}^k \right)
\end{aligned}
$$

$$
(MRLS) \quad s.t. \quad \sum_{i=1}^{T-1} x_{i\ell}^1 = d_\ell, \qquad \ell \in \mathbb{I}, \tag{3–63}
$$

$$
\sum_{i=t_1+1}^{T-1} x_{i\ell}^2 = x_{t_1\ell}^1, \qquad \ell \in \mathbb{I}, \tag{3–64}
$$

$$
\sum_{i=t_2+1}^{T-1} x_{i\ell}^3 = \sum_{k=1}^{2} x_{t_2\ell}^k, \qquad \ell \in \mathbb{I}, \tag{3–65}
$$

$$
\cdots\cdots
$$

$$
\sum_{i=t_\beta+1}^{T} x_{i\ell}^{\beta+1} = \sum_{k=1}^{\beta} x_{t_\beta\ell}^k, \quad \ell \in \mathbb{I}, \tag{3–66}
$$

$$
1 \le t_1 < t_2 < \cdots < t_\beta < T \text{ and integer}. \tag{3–67}
$$

We combine the enumeration of periods when the disruptions happen and the pre-processing algorithm to reformulate *MRLS* as a single stage linear programming problem. We extend the reformulation scheme in Section 3.4 for the single disruption case to the multiple disruption case. First, we use a two-disruption case as an example to explain our reformulation.

According to the problem setting, we have $t_2 > t_1$, where t_1 and t_2 represent the periods for the first and the second disruptions, respectively. The decision of $x_{i\ell}^2$ is based

on the period when the first disruption happens and $x^3_{i\ell}$ is based on the period when the first and the second disruptions happen. Thus, we let the production quantity be a function of the disruption history. For instance, we let $x^2_{i\ell}(t_1)$ and $x^3_{i\ell}(t_1, t_2)$ represent the extra production quantities after the first and the second disruptions, respectively. We use these as decision variables in the formulation. Finally, for a given t_1, we use $\theta^2(t_1)$ to represent the total extra cost after the second disruption. In this way, we enumerate all possible scenarios for the period when the second disruption happens and rewrite the formulation for the two-disruption case as follows:

$$\min_{x^1} \sum_{i=1}^{T}\sum_{\ell=1}^{T} c_{i\ell}x^1_{i\ell} + \max_{t_1} \min_{x^2(t_1)} \sum_{i=t_1+1}^{T}\sum_{\ell=1}^{T} c_{i\ell}x^2_{i\ell}(t_1) - \sum_{\ell=1}^{T} c_{t_1\ell}x^1_{t_1\ell} + \theta^2(t_1)$$

$$\text{s.t.} \quad \min_{x^3(t_1,t_2)} \sum_{i=t_2+1}^{T}\sum_{\ell=1}^{T} c_{i\ell}x^3_{i\ell}(t_1, t_2) - \sum_{\ell=1}^{T} c_{t_2\ell}(x^2_{t_2\ell}(t_1) + x^1_{t_2\ell}) \le \theta^2(t_1), \quad 1 \le t_1 < t_2 < T, \tag{3–68}$$

$$\sum_{i=t_2+1}^{T} x^3_{i\ell}(t_1, t_2) = x^2_{t_2\ell}(t_1) + x^1_{t_2\ell}, \quad \ell \in \mathbb{I}, 1 \le t_1 < t_2 < T, \tag{3–69}$$

Constraints (3 − −64).

The left hand of the above constraint can be simplified to the case without "min" operations. It results in a linear programming formulation. Because t_2 is the last disruption period, the extra production amount after t_2 to satisfy the unsatisfied demand for each period ℓ, $\ell = 1, \cdots, T$, $x^3_{i\ell}(t_1, t_2)$ is equal to $x^1_{t_2\ell} + x^2_{t_2\ell}(t_1)$. We can apply an idea similar to that used to solve the robust lot-sizing problem with single disruption and no setup cost case. We select a period after t_2 that provides the minimal unit cost to satisfy the unsatisfied demand in period ℓ due to the disruption in t_2. That is, we denote $m_{t_2\ell} = \min\{c_{i\ell}, \ i > t_2\}$ to be the minimum unit production cost for the unsatisfied demand in period ℓ. Constraint (3–68) can be rewritten as:

$$\sum_{i=t_2+1}^{T}\sum_{\ell=1}^{T} m_{t_2\ell}(x^2_{t_2\ell}(t_1) + x^1_{t_2\ell}) - \sum_{\ell=1}^{T} c_{t_2\ell}(x^2_{t_2\ell}(t_1) + x^1_{t_2\ell}) \le \theta^2(t_1), \quad 1 \le t_1 < t_2 \le T. \tag{3–70}$$

Now, we enumerate all possible scenarios of the period when the first disruption happens. We can reformulate the two-disruption cases as follows:

$$\min_{x^1,\theta^1} \sum_{i=1}^{T}\sum_{\ell=1}^{T} c_{i\ell}x^1_{i\ell} + \theta^1$$

$$s.t. \quad \min_{x^2(t_1),\theta^2(t_1)} \sum_{i=t_1+1}^{T}\sum_{\ell=1}^{T} c_{i\ell}x^2_{i\ell}(t_1) - \sum_{\ell=1}^{T} c_{t_1\ell}x^1_{t_1\ell} + \theta^2(t_1) \le \theta^1, \quad 1 \le t_1 < T, \tag{3–71}$$

Constraints (3–64), (3–69), (3–70).

We can claim that the above formulation is equivalent to the formulation without "min" operation on the left side of constraints (3–71) as shown in the following.

$$\sum_{i=t_1+1}^{T}\sum_{\ell=1}^{T} c_{i\ell}x^2_{i\ell}(t_1) - \sum_{\ell=1}^{T} c_{t_1\ell}x^1_{t_1\ell} + \theta^2(t_1) \le \theta^1, \quad 1 \le t_1 < T. \tag{3–72}$$

The reason lies in the following two facts:

In the optimal solution, for the formulation including (3–72), θ^1 should be the maximum value of the left side corresponding to a certain time period t. That is, there exists at least one tight inequality in (3–72) for θ^1.

In the optimal solution, for the formulation including (3–72), we have at least one tight inequality in which the left side achieves the minimum. Otherwise, θ^1 can be decreased and we have a contradiction. Note here, we do not need to consider the "min" operation for the non-tight inequalities, since it will not affect the optimal objective value.

Thus, we can substitute inequality (3–72) to (3–71). The two-disruption case can be solved by using the following linear programming formulation (MB-2):

$$\min_{x^1,\,\theta^1} \sum_{i=1}^{T}\sum_{\ell=1}^{T} c_{i\ell}x^1_{i\ell} + \theta^1$$

$$s.t. \quad \text{Constraints (3–64), (3–69), (3–70), (3–72).}$$

Now, we can analyze the structure of (MB-2) and extend it to the multiple disruptions case. For the two disruptions case, based on the enumeration of the scenarios of t_1, we introduce an artificial variable θ^1 and there are $\mathcal{O}(T)$ constraints of (3–64) and (3–72). For the second disruption, for a given scenario of t_1, we introduce an artificial variable $\theta^2(t_1)$. We also increase the dimension of the second stage decision variable $x^2_{i\ell}$ to be $x^2_{i\ell}(t_1)$. Therefore, there are $\mathcal{O}(T^3)$ constraints of (3–69) and $\mathcal{O}(T^2)$ constraints of (3–70).

Based on the similar idea, we can introduce the notation $x^k_{i\ell}(t_1, \dots, t_{k-1})$ to be the extra production quantity in period i to satisfy the demand in period ℓ after the t^{th}_{k-1} disruption. Then, we can observe that the following conclusion holds for multi-stage robust lot-sizing problems.

Theorem 3.2. *For the multi-stage robust lot-sizing problem with β disruptions and without setup cost, the problem can be formulated as the following single stage linear program.*

$$\min_{x^1,\theta^1} \sum_{i=1}^{T}\sum_{\ell=1}^{T} c_{i\ell}x^1_{i\ell} + \theta^1 \tag{3–73}$$

$$\text{s.t.} \quad \sum_{i=1}^{T} x^1_{i\ell} = d_\ell, \qquad \ell \in \mathbb{I} \tag{3–74}$$

$$\sum_{i=t_1+1}^{T} x^2_{i\ell}(t_1) = x^1_{t_1\ell}, \qquad \ell \in \mathbb{I} \tag{3–75}$$

$$\sum_{i=t_2+1}^{T} x^3_{i\ell}(t_1, t_2) = x^1_{t_2\ell} + x^2_{t_2\ell}(t_1), \qquad \ell \in \mathbb{I} \tag{3–76}$$

$$\cdots\cdots$$

$$\sum_{i=t_{\beta-1}+1}^{T} x^\beta_{i\ell}(t_1, \dots, t_{\beta-1}) = x^1_{t_{\beta-1}\ell} + \sum_{k=2}^{\beta-1} x^k_{t_{\beta-1}\ell}(t_1, \dots, t_{k-1}), \qquad \ell \in \mathbb{I} \tag{3–77}$$

$$\sum_{i=t_1+1}^{T}\sum_{\ell=1}^{T} c_{i\ell}x^2_{i\ell}(t_1) - \sum_{\ell=1}^{T} c_{t_1\ell}x^1_{t_1\ell} + \theta^2(t_1) \le \theta^1, \tag{3–78}$$

$$\sum_{i=t_2+1}^{T}\sum_{\ell=1}^{T} c_{i\ell}x^3_{i\ell}(t_1, t_2) - \sum_{\ell=1}^{T} c_{t_2\ell}(x^1_{t_2\ell} + x^2_{t_2\ell}(t_1)) + \theta^3(t_1, t_2) \le \theta^2(t_1), \tag{3–79}$$

$\cdots\cdots$

$$\sum_{i=t_{\beta-1}+1}^{T}\sum_{\ell=1}^{T} c_{i\ell}x^{\beta}_{i\ell}(t_1, \cdots, t_{\beta-1}) - \sum_{\ell=1}^{T} c_{t_{\beta-1}\ell}(x^1_{t_{\beta-1}\ell} + \sum_{k=2}^{\beta-1} x^k_{t_{\beta-1}\ell}(t_1, \cdots, t_{k-1}))$$
$$+\theta^{\beta}(t_1, \ldots, t_{\beta-1}) \le \theta^{\beta-1}(t_1, \cdots, t_{\beta-2}), \tag{3–80}$$

$$\sum_{\ell=1}^{T} m_{t_\beta\ell}(x^1_{t_\beta\ell} + \sum_{k=2}^{\beta} x^k_{t_\beta\ell}(t_1, \cdots, t_{k-1})) - \sum_{\ell=1}^{T} c_{t_\beta\ell}(x^1_{t_\beta\ell} + \sum_{k=2}^{\beta} x^k_{t_\beta\ell}(t_1, \cdots, t_{k-1}))$$
$$\le \theta^{\beta}(t_1, \cdots, t_{\beta-1}), \tag{3–81}$$

$$\theta^k \textit{ free; } x^k \ge 0, \ \forall 1 \le k \le \beta, \tag{3–82}$$

where $m_{t_\beta\ell} = \min\{c_{i\ell},\ i > t_\beta\}$ *and* $1 \le t_1 < t_2 < \cdots < t_\beta < T$.

Proof. We have shown that the conclusion holds for the cases in which there are one or two disruptions. For the multiple disruption case, we can explore the different combinations of $t_1, t_2, \ldots, t_\beta$ to a scenario tree with $\beta + 1$ depth. The root node indicates that there are no disruptions at the very beginning. The children of root node describe the possible first disruption time t_1. That is, there are $T - 1$ children of root node. Corresponding to each particular tree node t_1, there are $T - t_1 - 1$ children to represent the possible second disruption time period. For the general case, corresponding to each tree node t_k, there are $T - t_k - 1$ children to represent the possible $k + 1$ disruption time periods, based on the previous disruption realizations of $(t_1, t_2, \ldots, t_k)$ along the path from the root node to the tree node representing t_k. Leaf nodes with depth $\beta + 1$ represent the last disruption time period.

Therefore, in our approach, we enumerate all possible disruption combinations. For each tree node, corresponding to a particular realization of disruptions $(t_1, \ldots, t_k)$, we need to have the extra production in stage $k + 1$ to cover the unsatisfied demands due to disruption at t_k. Thus, constraints (3–74) to (3–77) hold.

Also, similar to the study for the two disruptions case, corresponding to the root node, we have

$$\min_{x^2,\theta^2} \sum_{i=1}^{T} \sum_{\ell=1}^{T} c_{i\ell} x^2_{i\ell}(t_1) - \sum_{\ell=1}^{T} c_{t_1\ell} x^1_{t_1\ell} + \theta^2(t_1) \leq \theta^1, \quad 1 \leq t_1 < T. \tag{3–83}$$

For each t_1, we can also consider inequality (3–83) corresponding to the link between root node and its child t_1.

For the general case, corresponding to each tree node $(t_1, t_2, \dots, t_k)$, except leaf nodes, we have

$$\begin{aligned} &\min_{x^{k+2},\theta^{k+2}} \sum_{i=t_{k+1}+1}^{T} \sum_{\ell=1}^{T} c_{i\ell} x^{k+2}_{i\ell}(t_1, \cdots, t_{k+1}) - \sum_{\ell=1}^{T} c_{t_{k+1}\ell}\Big(x^1_{t_{k+1}\ell} + \sum_{r=2}^{k+1} x^r_{t_{k+1}\ell}(t_1, \cdots, t_{r-1})\Big) \\ &+\theta^{k+2}(t_1, \dots, t_{k+1}) \leq \theta^{k+1}(t_1, \cdots, t_k), \quad t_k < t_{k+1} < T. \end{aligned} \tag{3–84}$$

Similarly, for a given t_{k+1}, we can consider the above inequality corresponding to the link between tree nodes $(t_1, \dots, t_k)$ and $(t_1, \dots, t_{k+1})$. Therefore, constraints (3–84) correspond to all links in the tree.

As shown in the proof for the two disruptions case, we can remove the "min" operations on the left side. That is, we can derive the corresponding constraints (3–78) to (3–81), which also correspond to all links in the tree. Then, we can obtain the following two conclusions.

(1) It can be observed that there exists at least one path from the root node to a leaf node such that, corresponding to each link along the path, the corresponding inequality in constraints (3–78) to (3–81) is tight. This path can be obtained by breadth-first search starting from the root node to find descendants along the links in which the constraints are tight (i.e., named tight links). If the breadth-first search terminates without reaching any leaf nodes, then we can reduce θ by a small positive value ϵ, which leads to a smaller objective value. Contradiction!

(2) It can also be observed that among all the candidate paths as shown in (1), there exists at least one path such that, for the inequality corresponding to each

link along the path, the left side achieves the minimum. We can also use the breadth-first search on the links among the paths that are tight. We check if the left side is tight for each tight inequality corresponding to each link. If the breadth-first search terminates without reaching any leaf node, then we can reduce the left side value and the θ value by a small positive value ϵ, which leads to a smaller objective value. Contradiction!

Based on (1) and (2), we will eventually find at least one path from the root node to a leaf node such that each inequality is tight corresponding to each link, and the left side for each tight inequality is minimized. The leaf node index gives the disruption periods in the optimal solution. Therefore, the conclusion holds. □

3.5.2 Setup Cost Case

In the above section, we show that the multi-stage robust lot-sizing problem can be reformulated as a single-stage deterministic equivalent linear programming problem with polynomial size of constraints and decision variables. For the case with setup cost in the first stage, we can similarly obtain the following formulation:

$$\min_{x^1,\theta^1} \sum_{i=1}^{T} f_i y_i + \sum_{i=1}^{T}\sum_{\ell=1}^{T} c_{i\ell} x^1_{i\ell} + \theta^1 \tag{3–85}$$

$$\text{s.t.} \quad (3\text{–}74) \text{ to } (3\text{–}81); \tag{3–86}$$

$$x^1_{i\ell},\ x^k_{i\ell}(t_1, \ldots, t_{k-1}) \le M y_i,\ \forall i, \ell \in \mathbb{I}, 1 \le k \le \beta. \tag{3–87}$$

3.6 Computational Results

In this section, we present the computational results to demonstrate the computational tractability of solution approaches we studied for the different cases of multi-stage robust lot-sizing problems. All computational experiments were carried out on a Linux workstation with a Pentium Dual 2.8G processor and 6G RAM. We use CPLEX 10.1 Callable Library to implement our algorithms and run the reformulated models.

3.6.1 Instance Generation

We generate instances based on different ratios of setup cost f_i to unit production cost p_i and different time horizons T. For the instances, we set the time horizon $T = 10$, 20, 30, 40 and 50, and the ratios of setup cost to unit production cost $f/p = 10$, 20, 30, and 40, respectively. There are 20 combinations in total.

For the cases with setup costs, corresponding to each of the combinations of T and f/p, we generate the random instances in which the unit production cost and the setup cost are uniformly distributed in the intervals as shown in Table 3-1.

For the cases without setup costs, corresponding to each of the combinations of T and p, we generate the random instances in which the unit production cost is uniformly distributed in the same interval as for the cases with setup costs shown in Table 3-1.

We also set the demand d_i, unit inventory cost h_i, and unit backlogging cost b_i uniformly distributed in the intervals $[500, 1000]$, $[5, 10]$, and $[10, 20]$ respectively.

3.6.2 Two-stage Robust Lot-sizing Problem

For two-stage robust lot-sizing problems, we test the case using outsourcing and backlogging to recover unfilled demands. The computational results for the robust lot-sizing problem with outsourcing and the robust lot-sizing problem with backlogging are illustrated in Section 3.6.2.1 and Section 3.6.2.2, respectively.

3.6.2.1 Two-stage Robust Lot-sizing Problem with Outsourcing

For this case, we test twenty combinations in which the time horizon $T = 10, 20, 30, 40,$ and 50 respectively and the ratio $f/p = 10, 20, 30,$ and 40, respectively. The computational results are shown in 3-2. Table 3-2 reports the computational results

Table 3-1. Parameter setting

	unit production cost p_i	setup cost f_i
ratio 10	[50, 100]	[500, 1000]
ratio 20	[50, 100]	[1000, 2000]
ratio 30	[40, 60]	[1000, 2000]
ratio 40	[40, 60]	[500, 1000]

for multiple disruption case in which we assume the number of disruption periods is 10% of the time horizon T. For each of the 20 combinations, we report the average values of 5 random instances. We report 1) the optimal objective value of the lot-sizing problem without disruptions, denoted as "SLS", 2) the objective value obtained by maximum pick heuristic, denoted as "MPSLS", 3) the objective value for the robust optimization formulation obtained by using CPLEX to solve the dual formulation (RDLS) for the outsourcing cost cases, and 4) the gap between (RDLS) and (MPSLS), denoted as "GAP(M::R)=(Obj_{MPSLS}-Obj_{RDLS})/(Obj_{RDLS})," and 5) the gap between (SLS) and (RDLS), denoted as "GAP(R::S)= (Obj_{RDLS}-Obj_{SLS})/(Obj_{SLS})." Compared with the maximum pick heuristics, the average gap is 55.8% for multiple disruption case. That means the total cost can be reduced by 55.8% by applying robust optimization approach (RDLS). Compared with the uncapacitated lot sizing problem without disruptions, the average gap between (SLS) and (RDLS) is 35.0%. That means total costs increase 35.0% by considering disruptions.

3.6.2.2 Two-stage Robust Lot-sizing Problem with a Single Disruption and Backlogging

First, we evaluate the efficiency of our formulation for lot-sizing with a single disruption and backlogging by comparing the total cost of our formulation with the no disruption case and maximum pickup heuristics. We test 20 combinations in which the time horizon $T = 10, 20, 30, 40,$ and 50 respectively and the ratio $f/p = 10, 20, 30,$ and 40, respectively. We report the performance of our formulation for lot-sizing with single disruption in Table 3.6.3. Corresponding to each combination, we report the value of 1) the objective value obtained by single item lot-sizing problem without the disruption, denoted as "SLS", 2)the objective value obtained by maximum pickup heuristics, denoted as "MP", 3) the objective value obtained by the lot-sizing with backlogging case, denoted as "SLS-B", 4) the gap between SLS and SLS-B, denoted as "GAP(B::S)=$(Obj_{SLS\text{-}B} - Obj_{SLS})/Obj_{SLS}$. 5) the gap between SLS-B and MP,

denoted as "GAP(B:M)=$(\text{Obj}_{\text{MP}} - \text{Obj}_{\text{SLS-B}})/\text{Obj}_{\text{SLS-B}}$". GAP(B::S) shows the growth of the total production with the consideration of setup cost. The average increasing rate is 11.92% for the robust lot-sizing with backlogging. GAP(B::M) shows the saving of the total production of our formulation for the lot-sizing problem with a single disruption and backlogging compared with the maximum pickup heuristics. The average saving rate are 24.55%.

The performance of the generated facet defining inequalities for the robust lot-sizing with backlogging is reported in Table 3.6.3. We test twelve combinations in which the time horizon $T = 100, 120$, and 140, respectively and the ratio $f/p = 10, 20, 30,$ and 40, respectively. For each combination, we run five instances and report the average performance of these five instances. Let "Cut" and "NoCut" denote the branch-and-cut algorithm with the facet-defining inequalities as cuts and the default CPLEX without adding facet defining inequalities. The testing instances for "Cut" and "NoCut" are same. We set the computational time limit as 1800 seconds for both cases. For the "NoCut" case, no instance can be finished within time limits. For the "Cut" case, all instances for $T = 100$ and $T = 120$ can be finished within 1800 seconds and achieve the optimal solution. The exact computational time and optimality gap are listed in Table 3.6.3.

3.6.3 Multi-stage Robust Lot-sizing Problem with Backlogging and without Setup Cost

Finally, we test multi-stage robust lot-sizing problems with backlogging, without setup cost, as described in Section 3.5. From Theorem 3.2, we can observe that the optimal solution for this type of problems can be obtained by solving a linear programming formulation that contains $\mathcal{O}(T^{\beta})$ constraints and $\mathcal{O}(T^{\beta})$ variables, where β is the number of total disruptions. Therefore, this formulation is pseudo-polynomial, in terms of time periods. We report the computational results in Tables 3-5 and 3-6 for two-disruption and three-disruption cases, respectively. For 2 disruption case, we solve a linear formulation that contains $\mathcal{O}(T^2)$ variables and constraints to get the optimal

solution. For the three-disruption case, we solve a linear formulation that contains $\mathcal{O}(T^3)$ variables and constraints to get the optimal solution. Similarly, we use the objective value obtained by the maximum pick heuristic as an upper bound to evaluate the cost savings by applying our robust optimization approach. It can be observed in Table 3-5 that, on average, our robust optimization approach has possible cost savings around 80.8%, compared with max pick heuristics. When compared with the uncapacitated lot-sizing problem without disruptions, our approach increases cost around 11.6%. It can be observed in Table 3-6 that, on average, our robust optimization approach has possible cost saving around 82.1%, compared with max pick heuristics. When compared with the uncapacitated lot-sizing problem without disruptions, our approach increases cost around 37.9%. Tables 3-5 and 3-6 also indicate that the computational time, which is reported in seconds, increases as the number of time periods and disruptions increases. But in general, the optimal solution can be obtained within one second for the two disruption case and within a minute for the three disruption case.

Table 3-2. Robust lot-sizing with outsourcing: multiple disruptions

		$T = 10$ Dis =1	$T = 20$ Dis =2	$T = 30$ Dis = 3	$T = 40$ Dis =4	$T = 50$ Dis =5
ratio=10	SLS	4.84×10^5	1.04×10^6	1.56×10^6	2.05×10^6	2.52×10^6
	MPSLS	8.37×10^5	1.94×10^6	3.16×10^6	3.96×10^6	5.58×10^6
	RDLS	6.53×10^5	1.39×10^6	2.06×10^6	2.72×10^6	3.37×10^6
	GAP(M::R)	28.2%	39.6%	53.4%	45.8%	65.6%
	GAP(R::S)	35.1%	33.0%	32.0%	32.6%	33.5%
ratio=20	SLS	5.17×10^5	1.11×10^6	1.51×10^6	2.07×10^6	2.55×10^6
	MPSLS	9.42×10^5	2.07×10^6	2.96×10^6	4.38×10^6	5.18×10^6
	RDLS	6.82×10^5	1.44×10^6	2.00×10^6	2.70×10^6	3.40×10^6
	GAP(M::R)	38.1%	44.0%	47.8%	62.0%	52.3%
	GAP(R::S)	31.9%	28.7%	32.4%	30.4%	33.5%
ratio=30	SLS	3.80×10^5	7.43×10^5	1.11×10^6	1.52×10^6	1.87×10^6
	MPSLS	6.40×10^5	1.45×10^6	2.11×10^6	2.83×10^6	3.81×10^6
	RDLS	5.15×10^5	1.03×10^6	1.54×10^6	2.09×10^6	2.59×10^6
	GAP(M::R)	24.2%	41.5%	37.3%	35.7%	47.5%
	GAP(R::S)	35.5%	38.0%	38.5%	37.7%	38.4%
ratio=40	SLS	3.75×10^5	7.60×10^5	1.14×10^6	1.51×10^6	1.87×10^6
	MPSLS	7.37×10^5	1.48×10^6	2.12×10^6	3.06×10^6	3.95×10^6
	RDLS	5.14×10^5	1.04×10^6	1.58×10^6	2.09×10^6	2.58×10^6
	GAP(M::R)	43.4%	42.7%	34.3%	46.6%	52.8%
	GAP(R::S)	37.0%	36.8%	38.7%	37.7%	38.3%

Table 3-3. Robust lot-sizing with backlogging: a single disruption

		T=10	T=20	T=30	T=40	T=50
ratio= 10	SLS	5.03×10^5	9.86×10^5	1.56×10^6	2.08×10^6	2.44×10^6
	MP	8.44×10^5	1.51×10^6	2.04×10^6	2.61×10^6	2.97×10^6
	SLS-B	5.38×10^5	1.17×10^6	1.69×10^6	2.28×10^6	2.73×10^6
	GAP(B::S)	7.05%	18.75%	8.35%	9.83%	11.88%
	GAP(B::M)	56.87%	29.06%	20.71%	14.47%	8.79%
ratio= 20	SLS	5.03×10^5	9.86×10^5	1.56×10^6	2.08×10^6	2.44×10^6
	MP	7.96×10^5	1.48×10^6	2.05×10^6	2.60×10^6	3.09×10^6
	SLS-B	5.89×10^5	1.20×10^6	1.72×10^6	2.32×10^6	2.83×10^6
	GAP(B::S)	17.02%	21.50%	10.50%	11.63%	16.09%
	GAP(B::M)	35.14%	23.33%	19.18%	12.07%	9.18%
ratio= 30	SLS	3.49×10^5	7.04×10^5	1.08×10^6	1.44×10^6	1.85×10^6
	MP	6.85×10^5	1.06×10^6	1.42×10^6	1.81×10^6	2.27×10^6
	SLS-B	4.48×10^5	8.28×10^5	1.13×10^6	1.51×10^6	2.02×10^6
	GAP(B::S)	22.09%	17.54%	5.20%	4.85%	9.19%
	GAP(B::M)	52.49%	28.02%	25.66%	19.86%	12.37%
ratio= 40	SLS	3.49×10^5	7.04×10^5	1.08×10^6	1.44×10^6	1.85×10^6
	MP	7.28×10^5	1.12×10^6	1.53×10^6	1.91×10^6	2.29×10^6
	SLS-B	4.70×10^5	8.81×10^5	1.15×10^6	1.56×10^6	2.09×10^6
	GAP(B::S)	25.74%	25.23%	6.05%	4.41%	12.97%
	GAP(B::M)	35.40%	27.12%	33.04%	22.43%	8.79%

Table 3-4. Robust lot-sizing with backlogging: branch-and-cut

	T=100				T=120				T=140			
	Cut		NoCut		Cut		NoCut		Cut		NoCut	
	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap
ratio=10	472	0%	1800	0.26%	1151	0%	1800	0.56%	1800	0.02%	1800	0.60%
ratio=20	576	0%	1800	0.49%	1273	0%	1800	0.61%	1800	0.04%	1800	0.65%
ratio=30	514	0%	1800	0.67%	1286	0%	1800	1.04%	1800	0.08%	1800	1.35%
ratio=40	644	0%	1800	0.98%	1168	0%	1800	1.21%	1800	0.11%	1800	1.20%

Table 3-5. Multi-stage robust lot-sizing problem with 2 disruptions

		$T = 10$		$T = 20$		$T = 30$		$T = 40$		$T = 50$	
		Obj	Time	Obj	Time	Obj	Time	Obj	Time	Obj	Time
ratio=10	SLS($\times 10^6$)	0.50	0.034	0.99	0.034	1.56	0.034	2.08	0.038	2.44	0.039
	MPSLS($\times 10^6$)	1.29	0.034	2.06	0.034	2.64	0.034	3.16	0.038	3.66	0.039
	NoSetup($\times 10^6$)	0.62	0.044	1.12	0.076	1.65	0.191	2.15	0.478	2.64	0.952
	Gap(M::N)	116.4%		89.9%		60.2%		47.1%		44.6%	
	Gap(N::S)	23.2%		13.6%		5.7%		3.5%		8.2%	
ratio=30	SLS($\times 10^6$)	0.35	0.034	0.70	0.034	1.08	0.036	1.44	0.038	1.87	0.038
	MPSLS($\times 10^6$)	1.20	0.034	1.63	0.034	1.93	0.036	2.42	0.038	3.03	0.038
	NoSetup($\times 10^6$)	4.45	0.041	8.03	0.074	1.19	0.209	1.55	0.555	1.91	1.153
	Gap(M::N)	169.2%		102.6%		62.9%		56.1%		58.7%	
	Gap(N::S)	27.5%		14.1%		10.3%		8.0%		2.1%	

Table 3-6. Multi-stage robust lot-sizing problem with 3 disruptions

		$T = 10$		$T = 20$		$T = 30$		$T = 40$		$T = 50$	
		Obj	Time	Obj	Time	Obj	Time	Obj	Time	Obj	Time
ratio=10	SLS($\times 10^6$)	0.50	0.034	0.99	0.034	1.56	0.034	2.07	0.038	2.44	0.039
	MPSLS($\times 10^6$)	1.53	0.034	2.44	0.034	3.09	0.034	3.69	0.038	4.24	0.039
	NoSetup($\times 10^6$)	0.73	0.065	1.35	0.557	2.01	3.066	2.62	10.165	3.26	32.688
	Gap(M::N)	147.8%		81.4%		53.5%		40.5%		30.2%	
	Gap(N::S)	45.4%		36.6%		29.0%		26.3%		33.8%	
ratio=30	SLS($\times 10^6$)	0.35	0.034	0.70	0.034	1.08	0.036	1.44	0.038	1.86	0.038
	MPSLS($\times 10^6$)	1.55	0.034	2.14	0.034	2.49	0.036	3.03	0.038	3.73	0.038
	NoSetup($\times 10^6$)	0.54	0.059	1.01	0.592	1.52	2.615	1.96	9.571	2.45	28.079
	Gap(M::N)	185.0%		112.9%		63.4%		54.6%		52.1%	
	Gap(N::S)	55.7%		42.9%		41.5%		36.7%		31.1%	

CHAPTER 4
STOCHASTIC LOT-SIZING PROBLEM WITH DETERMINISTIC DEMANDS AND WAGNER-WHITIN COSTS

4.1 Introduction

In practice, the cost parameters may be uncertain. This research studies cost parameter uncertainty and is motivated by tactical production decisions, not operational production decisions, for chemical companies. As typical supply chain characteristics of chemical industry, a chemical company produces mainly functional products, which are defined as ones that have a long product lifecycle and stable demand (see, e.g., Lee and Chen 2005). Under this situation, the demand is stable and easily forecasted accurately. However, the cost parameter forecasts may need to be adjusted monthly or quarterly. We can formulate this problem as a two-stage stochastic lot-sizing problem (SULS), in which we let a specific time period, e.g., time period p, represent the time for which the forecast needs to be adjusted. The cost parameters after the given time period will be uncertain and follow a discrete probability distribution. The detailed two-stage stochastic integer programming formulation can be described as follows (cf. Birge and Louveaux 1997; Louveaux and Schultz 2003):

$$
\begin{aligned}
\min \ & \sum_{i=1}^{p}(\alpha_i x_i + \beta_i' z_i + h_i' s_i) + E_\xi \mathcal{Q}(x, z, s, \xi(w)) & \\
s.t. \ & x_i + s_{i-1} = d_i + s_i, & 1 \le i \le p \\
& x_i \le M z_i, & 1 \le i \le p \\
& x_i, s_i \ge 0, \quad z_i \in \{0, 1\}, & 1 \le i \le p,
\end{aligned}
$$

where

$$
\mathcal{Q}(x, z, s, \xi(w)) = \min_{x^2, z^2, s^2} \left\{ \sum_{i=p+1}^{T} \begin{pmatrix} \alpha_i(w) x_i^2(w) \\ +\beta_i'(w) z_i^2(w) \\ +h_i'(w) s_i^2(w) \end{pmatrix} \middle| \begin{array}{ll} x_i^2(w) + s_{i-1}^2(w) = d_i + s_i^2(w), & p+1 \le i \le T \\ x_i^2(w) \le M z_i^2(w), & p+1 \le i \le T \\ x_i^2(w), s_i^2(w) \ge 0, z_i^2(w) \in \{0, 1\}, & p+1 \le i \le T \end{array} \right\}.
$$

Note, here $s^2_{i-1}(w) = s_p$ if $i-1 = p$. Decision variables (x_i, z_i, s_i) and $(x^2_i(\omega), z^2_i(\omega), s^2_i(\omega))$ represent the setup decision, and production and inventory levels on the first and the second stages, respectively. The corresponding cost parameters are (α, β', h') and $(\alpha(w), \beta'(w), h'(w))$.

For SULS, polynomial time algorithms (see, e.g., Guan and Miller 2008; Huang and Küçükyavuz 2008) and efficient cutting planes (see, e.g., Guan et al. 2006b; Summa and Wolsey 2006) have been studied recently for its deterministic equivalent scenario-tree based formulations. The reformulation for the problem was originally introduced in Ahmed et al. (2003). Later on, in Guan et al. (2006a), this reformulation was proved to be equivalent to adding the (ℓ, S) inequalities in the original formulation, in terms of providing LP lower bounds. However, both approaches could not provide integral solutions for SULS. An extended formulation that provides integral solutions for SULS up to now is only for two-period cases, which was developed in Guan et al. (2006a). To the best of our knowledge, there is no previous research on developing an extended formulation that provides integral solutions for multi-period SULS. This is another motivation for our research, and this paper contributes to the literature on deriving an extended formulation for multi-period SULS.

4.2 An Extended Formulation

After exploring possible realizations of the second stage random variables $(\alpha_i, \beta'_i, h'_i)$, we can generate a two-stage stochastic scenario tree for the problem as shown in Figure 4-1. Nodes $1, \cdots, p$ are first stage nodes and nodes $q_1, \cdots, \ell_W$ are second stage nodes, where branching node p connects the first and the second stages and is unique on the scenario tree. Assuming there are W possible scenarios, we let $\mathcal{P}(\omega)$, $1 \leq \omega \leq W$, represent the branch where scenario ω occurs and accordingly let ρ_ω, $1 \leq \omega \leq W$, represent the probability that scenario ω will occur. Since there is no demand uncertainty, we have $d_i = d_{t(i)} \geq 0$ for the second stage nodes, where $t(i)$ represents the time period of node i. We let $\mathcal{V}$ represent the set of nodes on the

scenario tree, and $\mathcal{V}(i)$ present the set of nodes which are descendants of node i (including node i itself). For each node $i \in \mathcal{V} \setminus \{1\}$, there exists a unique parent, denoted as node i^-. Finally, for each non-leaf node i, we let $\mathcal{C}(i)$ represent the set of its children. For our setting, $\mathcal{C}(i)$ contains a single element if the non-leaf node $i \neq p$. The deterministic equivalent formulation for two-stage SULS can be described as follows:

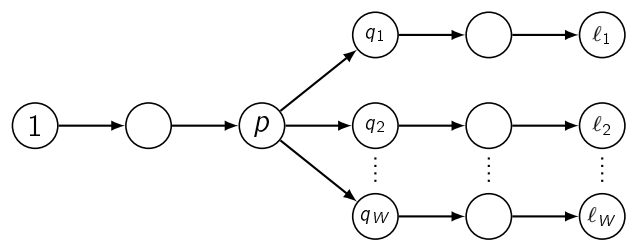


Figure 4-1. The scenario tree for two-stage SULS

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{p}(\alpha_i x_i + \beta_i' z_i + h_i' s_i) + \sum_{\omega=1}^{W} \rho_\omega \left[\sum_{i \in \mathcal{P}(\omega); t(i) \geq t(p)+1} (\alpha_i x_i + \beta_i' z_i + h_i' s_i) \right] \\
s.t. \quad & x_i + s_{i^-} = d_i + s_i, && i \in \mathcal{V}, \\
& x_i \leq M z_i, && i \in \mathcal{V}, \\
& x_i,\ s_i \geq 0,\ z_i \in \{0, 1\}, && i \in \mathcal{V},
\end{aligned}
$$

where x_i, s_i, and z_i present the production level, inventory level, and production set-up indicator in the state defined by node i whose corresponding time period is $t(i)$. Without loss of generality, we can assume $s_0 = 0$.

To define Wagner-Whitin costs for two-stage SULS, we substitute $x_i = d_i + s_i - s_{i^-}$ to eliminate the production variable x_i. Then, we obtain a reformulation of two-stage SULS in the $(s,\ z)$ space as follows:

$$
\begin{aligned}
& \sum_{i=1}^{p}(\alpha_i x_i + \beta_i' z_i + h_i' s_i) + \sum_{\omega=1}^{W} \rho_\omega \left[\sum_{i \in \mathcal{P}(\omega); t(i) \geq t(p)+1} (\alpha_i x_i + \beta_i' z_i + h_i' s_i) \right] \\
= \ & \sum_{i=1}^{p}[\alpha_i(d_i + s_i - s_{i-1}) + \beta_i' z_i + h_i' s_i] + \sum_{\omega} = 1^W \rho_\omega \left[\sum_{i \in \mathcal{P}(\omega); t(i) \geq t(p)+1} (\alpha_i(d_i + s_i - s_{i^-}) + \beta_i' z_i + h_i' s_i) \right]
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{i=1}^{p-1}(\alpha_i + h'_i - \alpha_{i+1})s_i + (\alpha_p + h'_p - \sum_{\omega=1}^{W}\rho_\omega\alpha_{q_\omega})s_p + \sum_{i=1}^{p}\beta'_i z_i + \sum_{i=1}^{p}\alpha_i d_i \\
&\quad + \sum_{\omega=1}^{W}\sum_{i\in\mathcal{P}(\omega);t(i)\geq t(p)+1}\rho_\omega(\alpha_i + h'_i - \alpha_{\mathcal{C}(i)})s_i + \sum_{\omega=1}^{W}\sum_{i\in\mathcal{P}(\omega);t(i)\geq t(p)+1}\rho_\omega\beta'_i z_i \\
&\quad + \sum_{\omega=1}^{W}\sum_{i\in\mathcal{P}(\omega);t(i)\geq t(p)+1}\rho_\omega\alpha_i d_i \\
&= \sum_{i\in\mathcal{V}}(h_i s_i + \beta_i z_i) + (\text{a constant}),
\end{aligned}
$$

where

$$
h_i = \begin{cases} \alpha_i + h'_i - \alpha_{i+1}, & 1 \leq i \leq p-1 \\ \alpha_p + h'_p - \sum_{\omega=1}^{W}\rho_\omega\alpha_{q_\omega}, & i = p \\ \rho_\omega(\alpha_i + h'_i - \alpha_{\mathcal{C}(i)}), & t(i) \geq t(p)+1, \end{cases} \quad \text{and} \quad \beta_i = \begin{cases} \beta'_i, & 1 \leq i \leq p \\ \rho_\omega\beta'_i, & t(i) \geq t(p)+1, \end{cases}
$$

where $\alpha_{\mathcal{C}(i)} = 0$ if $t(i) = T$.

Definition 1. *A two-stage SULS is said to have Wagner-Whitin costs if, for each $i \in \mathcal{V}$, we have*

$$
h_i \geq 0 \textit{ and } \quad \beta_i \geq \sum_{j\in\mathcal{C}(i)}\beta_j \textit{ with } \beta_j = 0 \textit{ if } j \in \mathcal{C}(i) \textit{ and } t(i) = T.
$$

Accordingly, we denote the problem satisfying the Wagner-Whitin cost setting as two-stage SULS-WW. A problem satisfying the Wagner-Whitin cost setting is also referred to as "without speculative motives" (see, Wagelmans et al. 1992, Pochet and Wolsey 1994, among others). Under this setting, we will not set up production at node $i \in \mathcal{V}$, if the inventory entering i is sufficient to satisfy the demand at i. The condition $h_i \geq 0$ is the same as the deterministic ULS Wagner-Whitin cost setting. Now, we provide a 4-period example in Figure 4-2, to indicate that $\beta_i \geq \sum_{j\in\mathcal{C}(i)}\beta_j$ is necessary to guarantee the Wagner-Whitin property.

As shown in Figure 4-2, for each node, the unit production cost is zero. The setup and the unit inventory costs are $(0,\ +\infty\ +\infty,\ 0,\ 0,\ +\infty)$ and $(100,\ 0,\ 0,\ 0,\ 0,\ 0)$,

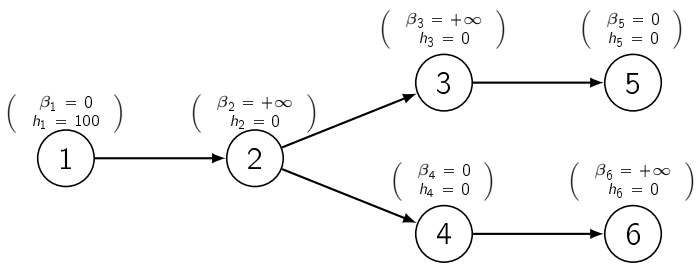


Figure 4-2. An example of two-stage SULS

respectively. Problem parameters satisfy the condition $h_i \geq 0$, but do not satisfy the condition $\beta_i \geq \sum_{j \in \mathcal{C}(i)} \beta_j$ in Definition 1. We assume the demand in each node is a non-zero finite number. In order to satisfy the demand at each time period and minimize the total cost, the productions are set up in nodes 1, 4, and 5. The production in node 1 covers the demands for nodes 1, 2, 3, and 4. Meanwhile, node 4 is set up to produce and satisfy the demand in node 6. Thus, there is inventory left from parent node and production set up for node 4 simultaneously. The Wagner-Whitin property for the deterministic case, $x_i s_{i^-} = 0$, does not hold here.

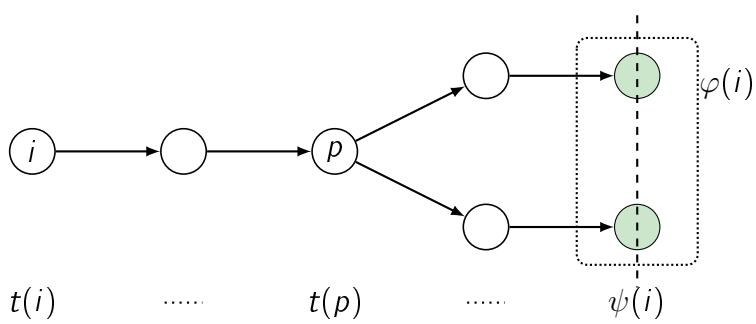


Figure 4-3. The subtree of node i

Now, we consider the optimal solution form of inventory level s_i for each $i \in \mathcal{V}$. Let $W(i)$ represent the set of scenarios that will occur after node i. For instance, if $i \leq p$, $W(i) = \{1, \ldots, W\}$; otherwise, if $t(i) \geq t(p) + 1$, $W(i)$ contains a single element ω such that $i \in \mathcal{P}(\omega)$. Let $\psi(i) = \min\{t : [1 - \sum_{j \in \mathcal{P}_t^i(\omega) \setminus \{i\}} z_j]^+ = 0,\ \omega \in W(i)\}$ if there exists a $k \in \mathcal{P}_T^i(\omega) \setminus \{i\},\ \omega \in W(i)$ such that $z_k = 1$, and $\psi(i) = T + 1$ o.w., as shown in Figure 4-3, where $\mathcal{P}_t^i(\omega)$ represents the path from node i to its descendant node at time

period t on the branch corresponding to scenario ω. Let $\varphi(i) = \bigcup_{\omega \in W(i)} \arg\min\{t(k) : k \in \mathcal{P}_T^i(\omega) \setminus \{i\} \text{ and } z_k = 1\}$. From observation, we see that the optimal inventory level s_i covers the demands at time periods after $t(i)$ and before $\psi(i)$. In the following proposition, we provide the closed form of inventory level s_i, $i \in \mathcal{V}$:

Proposition 4.1. *For two-stage SULS-WW, corresponding to each node $i \in \mathcal{V}$, there exists an optimal solution in the form*

$$s_i = \sum_{t=t(i)+1}^{\psi(i)-1} d_t, \ z_j = 1 \textit{ if } t(j) = \psi(i), \textit{ and } z_j = 0 \textit{ if } t(i) < t(j) < \psi(i).$$

Proof. Let $\Lambda = \{i \in \mathcal{V} : z_i = 1\}$. For each $i \in \mathcal{V}$, the conclusion is obvious if $\varphi(i) = \emptyset$. Now, we assume $\varphi(i) \neq \emptyset$ and let $i^* = \arg\max\{t(j) : j \in \varphi(i)\}$. We prove this proposition by two claims:

Claim 1: For each $i \in \Lambda$, $s_i = \sum_{t=t(i)+1}^{t(i^*)-1} d_t$.

Claim 2: Corresponding to each $i \in \Lambda$, $z_j = 1$ if $j \in \mathcal{V}(i)$ and $t(j) = t(i^*)$; $z_j = 0$ if $j \in \mathcal{V}(i)$ and $t(i) < t(j) < t(i^*)$.

Proof of Claim 1: First, we have $s_i \geq \sum_{t=t(i)+1}^{t(i^*)-1} d_t$ in order to satisfy the demands along each path $\mathcal{P}_{t(i^*)}^i(\omega) \setminus \{i, i^*\}$, $\omega \in W(i)$. If $s_i > \sum_{t=t(i)+1}^{t(i^*)-1} d_t$, for instance, $s_i = \sum_{t=t(i)+1}^{t(i^*)-1} d_t + \varepsilon$ for a small positive number $\varepsilon > 0$. Then, due to the fact that $h_i \geq 0$, we can reduce the production at i by ε and increase the production at each j, $j \in \varphi(i)$ by ε, which leads to a non-larger total cost. The problem is still feasible, which is a contradiction; thus, Claim 1 holds.

Proof of Claim 2: Based on Claim 1, we have $s_i = \sum_{t=t(i)+1}^{t(i^*)-1} d_t$. Since $\beta_i \geq \sum_{j \in \mathcal{C}(i)} \beta_j$, setting up at nodes in the set $\{j : j \in \mathcal{V}(i) \text{ and } t(j) = t(i^*)\}$ instead of at nodes in the set $\{j : j \in \mathcal{V}(i) \text{ and } t(i) < t(j) < t(i^*)\}$ leads to a non-larger total cost. Therefore, there exists an optimal solution such that $z_j = 1$ if $j \in \mathcal{V}(i)$ and $t(j) = t(i^*)$; $z_j = 0$ if $j \in \mathcal{V}(i)$ and $t(i) < t(j) < t(i^*)$. Thus, Claim 2 holds.

Based on Claims 1 and 2, the proposition holds. □

Proposition 4.1 provides a stronger claim, as compared to the production path property for SULS described in Guan and Miller (2008), developed for general cost and demand settings. Under these general settings, we cannot guarantee that the Wagner-Whitin cost setting holds. The example in Figure 4-2 can still happen, and $x_i s_{i^-} = 0$ does not hold. That is, we can not guarantee $z_j = 1$ if $j \in \mathcal{V}(i)$ and $t(j) = \psi(i)$, and $z_j = 0$ if $j \in \mathcal{V}(i)$ and $t(i) < t(j) < \psi(i)$, as described in Proposition 4.1.

Let binary decision variable δ_t^i represent whether the inventory left at node i covers the demand at time period t, $t > t(i)$. If yes, then $\delta_t^i = 1$; otherwise, $\delta_t^i = 0$. Then, based on Proposition 4.1, the following three types of inequalities are valid for two-stage SULS-WW.

1. Path I inequalities: these inequalities are for the second stage nodes. For a given node i, $t(i) \geq t(p) + 1$, ω is determined and

$$\delta_t^i \geq 1 - \sum_{j \in \mathcal{P}_t^i(\omega) \setminus \{i\}} z_j, \qquad t \geq t(i) + 1. \tag{4–1}$$

That is, node i covers demands along the branch it belongs to until the next production cycle.

2. Path II inequalities: these inequalities are for the first stage nodes (except node p). For a given node i, $1 \leq t(i) \leq t(p) - 1$, we have

$$\delta_t^i \geq 1 - z_{i+1}, \qquad t = t(i) + 1, \tag{4–2}$$

$$\delta_t^i \geq \delta_t^{i+1} - z_{i+1}, \qquad t \geq t(i) + 2. \tag{4–3}$$

For the first stage node i (except node p), node $i + 1$ is its child. Then, (4–2) holds. Inequality (4–3) indicates that if node $i + 1$ is not set up and the inventory left from node $i + 1$ covers the demand at time period t, $t \geq t(i) + 2$, then the inventory from node i also covers the demand at time period t.

3. Connection inequalities: these inequalities are for the branching node p. Corresponding to p and each $q_\omega \in \mathcal{C}(p), \omega \in W(p)$, we have

$$\delta_t^p \geq 1 - z_{q_\omega}, \quad t = t(p) + 1, \tag{4–4}$$

$$\delta_t^p \geq \delta_t^{q_\omega} - z_{q_\omega}, \quad t \geq t(p) + 2. \tag{4–5}$$

Constraints (4–4) and (4–5) are similar to (4–2) and (4–3). Along each scenario path, if there is no setup in time period $t(p) + 1$, the inventory left at node p covers demands up to the same time period as the inventory left at the node in time period $t(p) + 1$ does.

Knowing δ_t^i, the inventory level at each node in the scenario tree is as follows:

$$s_i = \sum_{t=t(i)+1}^{T} d_t \delta_t^i, \qquad i \in \mathcal{V}. \tag{4–6}$$

Inequalities (4–1)-(4–5) and equation (4–6), assuming binary δ_t^i and z_i, guarantee the feasibility of the reformulation for two-stage SULS-WW because the demand at each time period will be covered.

Theorem 4.1. *The linear program (4–1)–(4–6) plus* $0 \leq \delta_t^i \leq 1$ *and* $0 \leq z_i \leq 1$, $i \in \mathcal{V}$, $t(i) + 1 \leq t \leq T$ *provides an extended formulation for two-stage SULS-WW.*

Proof. Constraint (4–6) can be put directly in the objective function. We prove this theorem by showing that the constraint matrix for Constraints (4–1) to (4–5) is totally unimodular.

To prove that the constraint matrix for constraints (4–1) to (4–5) is totally unimodular, we order variables z_i and δ_t^i with an outer loop i ranging from 1 to $|\mathcal{V}|$, and an inner loop t ranging from $t(i) + 1$ to T. Table 4-1 describes the constraint matrix corresponding to the example in Figure 4-2.

Table 4-1. The matrix of constraints (4–1) to (4–5) for the example in Figure 4-2

	z_1	z_2	z_3	z_4	z_5	z_6	δ_2^1	δ_3^1	δ_4^1	δ_3^2	δ_4^2	δ_4^3	δ_4^4
1					1							1	
2						1							1
3		1					1						
4		1						1		-1			
5		1							1		-1		
6			1							1			
7				1						1			
8			1								1	-1	
9				1							1		-1

As the submatrix corresponding to variables δ_t^i, $t(i) \geq t(p) + 2$, is an identity matrix, we need only consider the constraint submatrix for the rest variables, denoted as matrix A.

We show that for any column subset J of matrix A, there exists a partition J_1 and J_2 of J such that

$$\left| \sum_{j \in J_1} a_{ij} - \sum_{j \in J_2} a_{ij} \right| \leq 1, \text{ for each row } i. \tag{4–7}$$

We partition δ and z variables in J, starting from branching node p, and then extend it in both directions for nodes after p and before p, respectively.

Step 1. Allocate δ_t^p to J_1, z_p to J_1, and z_{q_ω} to J_2, where $t \geq t(p) + 1$, $q_\omega \in \mathcal{C}(p)$.

Step 2. Allocate $\delta_t^{q_\omega}$, $t \geq t(p) + 2$, to the same set as z_{q_ω} (if $z_{q_\omega} \in J$), or to the same set as δ_t^p (if $z_{q_\omega} \notin J$ and $\delta_t^p \in J$), or to J_1(if $z_{q_\omega} \notin J$ and $\delta_t^p \notin J$).

Step 3. Allocate z_i, $t(i) \geq t(p) + 2$, to the opposite set of $z_{M(i)}$, if there exists $z_{M(i)}$ in J, where $M(i)$ is the closest ancestor of node i, at or after time period $t(p)$. Otherwise, allocate z_i to the opposite set of δ_t^p if $\delta_t^p \in J$. If $\delta_t^p \notin J$, then allocate z_i to J_2.

Step 4. Allocate z_i, $1 \leq i \leq q - 1$, to the opposite set of $z_{m(i)}$, if there exists $z_{m(i)}$ in J, where $m(i)$ is the closest descendant of node i at or before time period $t(p)$. Otherwise, allocate z_i to J_1.

Step 5. Allocate δ_t^i, $1 \leq i \leq q-1$, to the opposite set as $z_{m(i)}$, if there exists $z_{m(i)}$ as shown in Step 4 in J. Otherwise, allocate δ_t^i to J_1.

Now, we verify that (4–7) holds for constraints (4–1) to (4–5) under the above partition. Corresponding to each row, if J contains at most one decision variable in A, then it is clear that (4–7) holds. In the following, we consider the case where J contains at least two decision variables in A:

1. For constraint (4–4), if $\{\delta_t^p, z_{q_\omega}\} \subseteq J$, δ_t^p and z_{q_ω} go to J_1 and J_2, respectively, based on Step 1. Thus, (4–7) holds.

2. For constraint (4–5), we only need to consider the following four cases:

2-1. $\{\delta_t^p, z_{q_\omega}\} \subseteq J$. The argument is the same as constraint (4–4).

2-2. $\{\delta_t^{q_\omega}, z_{q_\omega}\} \subseteq J$. $\delta_t^{q_\omega}$ and z_{q_ω} go to the same set based on Step 2.

2-3. $\{\delta_t^p, \delta_t^{q_\omega}\} \subseteq J$. Because $z_{q_\omega} \notin J$, δ_t^p and $\delta_t^{q_\omega}$ go to the same set based on Step 2.

2-4. $\{\delta_t^p, \delta_t^{q_\omega}, z_{q_\omega}\} \subseteq J$. δ_t^p and z_{q_ω} go to J_1 and J_2, respectively, based on Step 1. Then, (4–7) holds no matter the destination of $\delta_t^{q_\omega}$.

3. For constraint (4–1), we consider two types of constraints: where $t(i) \geq t(p)+2$ or $t(i) = t(p)+1$.

3-1. $t(i) \geq t(p)+2$. For this case, first of all, we need not consider δ_t^i based on the identity matrix argument at the beginning of the proof. Then, based on Step 3, z_i is assigned alternatively to J_1 and J_2. Thus, (4–7) holds.

3-2. $t(i) = t(p)+1$. For this case, we should consider $\delta_t^{q_\omega}$ since $t(q_\omega) = t(p)+1$. We discuss the cases where $\delta_t^{q_\omega} \notin J$ and $\delta_t^{q_\omega} \in J$.

3-2-1. $\delta_t^{q_\omega} \notin J$. The argument is the same as 3-1.

3-2-2. $\delta_t^{q_\omega} \in J$. Based on Steps 2 and 3, we see that $\delta_t^{q_\omega}$ is in the opposite subset of z_{j^*}, where j^* is the closest descendant of q_ω, because $\delta_t^{q_\omega}$ goes to the same set as z_{q_ω} (if $z_{q_\omega} \in J$), or δ_t^p (if $z_{q_\omega} \notin J$ and $\delta_t^p \in J$), or J_1 (if $\delta_t^p \notin J$ and $z_{q_\omega} \notin J$). Accordingly, z_{j^*} is in the opposite set of z_{q_ω}, or

δ_t^p, or J_1. We also observe that z_j, $j \in (\mathcal{P}_t^{q_\omega}(\omega) \setminus \{q_\omega\}) \cap J$, is assigned alternatively to J_1 and J_2 based on Step 3. Therefore, (4–7) holds.

4. For Constraint (4–2), we only need to consider the case where both δ_t^i and z_{i+1} are in J. Based on Step 5, δ_t^i goes to the opposite set of z_{i+1}, since z_{i+1} is the closest descendant of node i. Thus, (4–7) holds.

5. For Constraint (4–3), we discuss the following four cases:

5-1. $\{\delta_t^i, z_{i+1}\} \subseteq J$ and $\delta_t^{i+1} \notin J$. The argument is the same as for Constraint (4–2).

5-2. $\{\delta_t^{i+1}, z_{i+1}\} \subseteq J$ and $\delta_t^i \notin J$. δ_t^{i+1} and z_{i+1} go to the same set because both δ_t^{i+1} and z_{i+1} are in the opposite set of $z_{m(i+1)}$ (if $z_{m(i+1)} \in J$) or in J_1 (if $z_{m(i+1)} \notin J$), based on Steps 4 and 5.

5-3. $\{\delta_t^i, \delta_t^{i+1}\} \subseteq J$ and $z_{i+1} \notin J$. Because $z_{i+1} \notin J$, both δ_t^i and δ_t^{i+1} are in the opposite set of $z_{m(i+1)}$ (if $z_{m(i+1)} \in J$) or in J_1 (if $z_{m(i+1)} \notin J$), based on Step 5.

5-4. $\{\delta_t^i, \delta_t^{i+1}, z_{i+1}\} \subseteq J$. Based on Step 5, δ_t^i goes to the opposite set of z_{i+1}, since z_{i+1} is the closest descendant of node i. Then, (4–7) holds no matter the destination of δ_t^{i+1}.

Therefore, the desired property (4–7) holds for constraints (4–1) to (4–5), and the matrix for constraints (4–1) to (4–5) is totally unimodular. □

4.3 An Integral Polyhedron in the Original Space

Now, we study the integral polyhedron in (s, z) space. First, we introduce Q_o as a polyhedron in (s, z) space described as follows:

$$Q_o = \left\{ (s, z) : s_i \geq \sum_{t=t(i)+1}^{\tau} d_t \left(1 - \sum_{j \in \mathcal{P}_t^i(\omega_t) \setminus \{i\}} z_j \right), 0 \leq z_i \leq 1,\ s_i \geq 0, t(i) + 1 \leq \tau \leq T,\ \omega_t \in W(i) \right\}. \tag{4–8}$$

We prove that Q_o is an integral polyhedron for two-stage SULS-WW by showing that it is a projection of Q_r in $(s,\ z)$ space, where Q_r represents the polyhedron of two-stage SULS-WW in (s, z, δ) space, i.e., $Q_r = \{(s, z, \delta) | (s, z, \delta)$ satisfies (4–1) to (4–6)$\}$.

Theorem 4.2. *Q_o is an integral polyhedron for two-stage SULS-WW.*

Proof. We prove this theorem by showing that $Q_o = \text{Proj}_{(s,z)} Q_r$ in the following two claims:

Claim 1: Inequalities in Q_o are valid for Q_r. To prove Claim 1, we only need to show that

$$s_i \geq \sum_{t=t(i)+1}^{T} d_t[1 - \sum_{j \in \mathcal{P}_t^i(\omega_t)\setminus\{i\}} z_j]^+, \qquad \omega_t \in W(i), \tag{4–9}$$

because for given ω_t, $t(i)+1 \leq t \leq T$, we have

$$\sum_{t=t(i)+1}^{T} d_t[1 - \sum_{j \in \mathcal{P}_t^i(\omega_t)\setminus\{i\}} z_j]^+ \geq \sum_{t=t(i)+1}^{\tau} d_t(1 - \sum_{j \in \mathcal{P}_t^i(\omega_t)\setminus\{i\}} z_j), \; t(i)+1 \leq \tau \leq T.$$

We prove (4–9) by discussing two cases: (1) $t(i) \geq t(p)+1$ and (2) $t(i) \leq t(p)$.

For $t(i) \geq t(p)+1$, based on (4–6), (4–1), and nonnegativity of δ_t^i, we have

$$\begin{aligned} s_i &= \sum_{t=t(i)+1}^{T} d_t \delta_t^i \\ &\geq \sum_{t=t(i)+1}^{T} d_t \max\left\{0, 1 - \sum_{j \in \mathcal{P}_t^i(\omega_t)\setminus\{i\}} z_j\right\} \\ &= \sum_{t=t(i)+1}^{T} d_t \left[1 - \sum_{j \in \mathcal{P}_t^i(\omega_t)\setminus\{i\}} z_j\right]^+, \end{aligned}$$

where ω_t is the single element in $W(i)$ for each $t(i)+1 \leq t \leq T$. Then, (4–9) holds.

For $t(i) \leq t(p)$, we have

$$\begin{aligned} s_i &= \sum_{t=t(i)+1}^{T} d_t \delta_t^i = d_{t(i)+1}\delta_{t(i)+1}^i + \sum_{t=t(i)+2}^{T} d_t \delta_t^i \\ &\geq d_{i+1}[1 - z_{i+1}]^+ + \sum_{t=t(i)+2}^{T} d_t \left[\delta_t^{i+1} - z_{i+1}\right]^+ \text{ if } t(i) < t(p) \qquad (4\text{–}10) \\ \text{or } &\geq d_{q_{\omega_{t(i)+1}}}[1 - z_{q_{\omega_{t(i)+1}}}]^+ + \sum_{t=t(i)+2}^{T} d_t \left[\delta_t^{q_{\omega_t}} - z_{q_{\omega_t}}\right]^+ \text{ if } t(i) = t(p), \qquad (4\text{–}11) \end{aligned}$$

where (4–10) is based on (4–2), (4–3), and nonnegativity of δ_t^i, while (4–11) is based on (4–4), (4–5), and nonnegativity of δ_t^i. We also notice that if $t(i) = t(p)$, as shown in (4–11), $\left[\delta_t^{q_{\omega_t}} - z_{q_{\omega_t}}\right]^+ \geq \left[1 - \sum_{j \in \mathcal{P}_t^i(\omega_t)\setminus\{i\}} z_j\right]^+$ based on (4–1). Then, (4–9) holds for the $t(i) = t(p)$ case.

Then, we only need to show that if $t(i) < t(p)$,

$$\left[\delta_t^{i+1} - z_{i+1}\right]^+ \geq \left[1 - \sum_{j \in \mathcal{P}_t^i(\omega_t)\setminus\{i\}} z_j\right]^+ . \tag{4–12}$$

It is easy to see that (4–12) holds based on (4–2) and (4–3) if $t \leq t(p)$.

If $t \geq t(p) + 1$, because $\delta_t^j \geq \delta_t^{j+1} - z_{j+1}$, $i + 1 \leq j \leq p - 1$, holds based on (4–3), we have

$$\left[\delta_t^{i+1} - z_{i+1}\right]^+ \geq \left[\delta_t^p - \sum_{j=i+1}^{p} z_j\right]^+ .$$

If $t = t(p) + 1$, then

$$\left[\delta_t^p - \sum_{j=i+1}^{p} z_j\right]^+ \geq \left[1 - \sum_{j=i+1}^{p} z_j - z_{q_{\omega_t}}\right]^+ = \left[1 - \sum_{j \in \mathcal{P}_t^i(\omega_t)\setminus\{i\}} z_j\right]^+ ,$$

where the inequality follows from (4–4).

If $t > t(p) + 1$, then

$$\left[\delta_t^p - \sum_{j=i+1}^{p} z_j\right]^+ \geq \left[\delta_t^{q_{\omega_t}} - \sum_{j \in \mathcal{P}_{t(p)+1}^i(\omega_t)\setminus\{i\}} z_j\right]^+ \geq \left[1 - \sum_{j \in \mathcal{P}_t^i(\omega_t)\setminus\{i\}} z_j\right]^+ ,$$

where the first inequality follows from (4–5) and the second follows from (4–1); therefore, (4–12) holds and thus, (4–9) and Claim 1 hold.

Claim 2: For any extreme point $(s, z) \in Q_o$, we can construct δ such that $(s, z, \delta) \in Q_r$. That is, (s, z, δ) satisfies constraints (4–1) to (4–6). Now, for a given extreme point $(\widehat{s}, \widehat{z}) \in Q_o$, we let

$$\widehat{\delta}_t^i = \max_{\omega \in W(i)} \left[1 - \sum_{j \in \mathcal{P}_t^i(\omega)\setminus\{i\}} \widehat{z}_j\right]^+ . \tag{4–13}$$

Since $(\widehat{s}, \widehat{z})$ is an extreme point in Q_o, we first observe that $(\widehat{s}, \widehat{z})$ satisfies equation (4–6) based on (4–8) and (4–13). We also observe that (4–1), (4–2), and (4–4) hold, which directly follows from (4–13).

For (4–3), let ω^* be the scenario where $\widehat{\delta}_t^{i+1}$ achieves the maximum value. Then, it is easy to observe that ω^* is also the scenario where $\widehat{\delta}_t^{i}$ achieves the maximum. Then, according to (4–13),

$$\begin{aligned}
\widehat{\delta}_t^{i} &= \left[1 - \sum_{j \in \mathcal{P}_t^i(\omega^*) \setminus \{i\}} \widehat{z}_j\right]^+ \\
&\geq \left[1 - \sum_{j \in \mathcal{P}_t^i(\omega^*) \setminus \{i, i+1\}} \widehat{z}_j\right]^+ - \widehat{z}_{i+1} \\
&= \widehat{\delta}_t^{i+1} - \widehat{z}_{i+1}.
\end{aligned}$$

Thus, (4–3) holds.

For (4–5), according to (4–13),

$$\begin{aligned}
\widehat{\delta}_t^{p} &= \max_{\omega \in W(p)} \left[1 - \sum_{j \in \mathcal{P}_t^p(\omega) \setminus \{p\}} \widehat{z}_j\right]^+ \\
&\geq \left[1 - \sum_{j \in \mathcal{P}_t^p(\omega) \setminus \{p\}} \widehat{z}_j\right]^+, \text{ for each } \omega \in W(p).
\end{aligned}$$

Now, for each $\omega \in W(p)$,

$$\begin{aligned}
& \left[1 - \sum_{j \in \mathcal{P}_t^p(\omega) \setminus \{p\}} \widehat{z}_j\right]^+ \\
= & \left[1 - \sum_{j \in \mathcal{P}_t^p(\omega) \setminus \{p, q_\omega\}} \widehat{z}_j - \widehat{z}_{q_\omega}\right]^+ \\
= & \left[\widehat{\delta}_t^{q_\omega} - \widehat{z}_{q_\omega}\right]^+ \qquad (4\text{–}14) \\
\geq & \ \widehat{\delta}_t^{q_\omega} - \widehat{z}_{q_\omega},
\end{aligned}$$

where (4–14) follows from the fact that $\mathcal{P}_t^p(\omega) \setminus \{p, q_\omega\} = \mathcal{P}_t^{q_\omega}(\omega) \setminus \{q_\omega\}$ and $\widehat{\delta}_t^{q_\omega} = 1 - \sum_{j \in \mathcal{P}_t^{q_\omega}(\omega) \setminus \{q_\omega\}} \widehat{z}_j$. Then, (4–5) holds. Thus, $(\widehat{s}, \widehat{z}, \widehat{\delta}) \in Q_r$, and Claim 2 holds.

Therefore, the conclusion holds. □

We can observe that with Wagner-Whitin costs, the explicit formulation of the polyhedral description for two-stage SULS in the original space is described by $\mathcal{O}(|\mathcal{V}|)$ variables and $\mathcal{O}(TW|\mathcal{V}|)$ constraints, where $|\mathcal{V}|$ is the cardinality of $\mathcal{V}$.

4.4 Extensions

A part of our results can be applied to a more general multi-stage stochastic programming setting, which can address further uncertainties in periods $p+1, p+2, \ldots, T$. Under the multi-stage setting, it can be observed that Proposition 4.1 still holds, based on the Wagner-Whitin cost setting defined in Definition 1. Accordingly, we can obtain a reformulation similar to constraints (4–1) to (4–6). For instance, we have constraints (4–1) to (4–3) for the last and the first stage nodes, and equation (4–6) for the inventory level expression. For the nodes between the first and the last stages, constraints similar to (4–4) and (4–5) are valid (e.g., $\delta_t^i \geq 1 - z_k$, $k \in \mathcal{C}(i)$, $t = t(i)+1$ and $\delta_t^i \geq \delta_t^k - z_k$, $k \in \mathcal{C}(i)$, $t \geq t(i)+2$). Therefore, we can obtain a similar reformation with binary δ_t^i and z_i for the multi-stage case. However, it is unknown if the reformulation can provide an extended formulation that provides integral solutions for multi-stage SULS; this possibility is currently under investigation.

CHAPTER 5
STOCHASTIC LOT-SIZING PROBLEM WITH DETERMINISTIC DEMANDS AND BACKLOGGING

5.1 Introduction

Pochet and Wolsey (1988) provided the first polyhedral study of the uncapacitated lot-sizing problem with backlogging and the convex hull description for the problem was recently studied by Küçükyavuz and Pochet (2007). In addition, for the uncapacitated lot-sizing problem with start-up cost, van Hoesel et al. (1994) presented an extended formulation and an $\mathcal{O}(T^2)$ time separation algorithm. For the Wagner-Whitin costs case, Wagelmans et al. (1992) introduced the Wagner-Whitin costs, i.e., $\alpha_i + h'_i \geq \alpha_{i+1}$ for all time period $1 \leq i \leq T-1$,, where α_i and h'_i are the unit production and inventory costs for time period i, and implemented an $\mathcal{O}(T)$ time dynamic programming algorithm to solve ULS with Wagner-Whitin costs. For the Wagner-Whitin cost case, Pochet and Wolsey (1994) generated an explicit formulation of convex hull for ULS with backlogging with $\mathcal{O}(2^T)$ constraints and an $\mathcal{O}(T^2 \log T)$ time separation algorithm.

The deterministic equivalent formulation for two-stage SULS with backlogging and Wagner-Whitin costs can be described as follows:

$$
\text{(SULSB-WW)} \quad
\begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{p}(\alpha_i x_i + \beta'_i z_i + h'_i s_i + b'_i \ell_i) + \sum_{w=1}^{W} \rho_w \left[\sum_{i \in \mathcal{P}_w, t(i) \geq t(p)+1} (\alpha_i\, x_i + \beta'_i z_i + h'_i s_i + b'_i \ell_i) \right] & \\
s.t. & x_i + s_{i^-} + \ell_i = d_i + s_i + \ell_{i^-}, & i \in \mathcal{V} \\
& x_i \leq M_i z_i, & i \in \mathcal{V} \\
& x_i,\ s_i,\ \ell_i \geq 0,\ z_i \in \{0,1\}, & i \in \mathcal{V},
\end{array}
$$

where x_i, s_i, ℓ_i, and z_i present the production level, inventory level, backlogging level, and production set-up indicator in the state defined by node i whose corresponding period is $t(i)$. Node i^- is the parent node of node i. Without loss of generality, we can assume $s_0 = 0$, $\ell_i = 0$, and tighten $M_i = \sum_{k=t(i)}^{T} d_k$, where $i \in \mathcal{L}$

In the remaining part of this chapter, we study two-stage stochastic ULS with backlogging, Wagner-Whitin costs and deterministic demands, denoted as two-stage SULSB-WW. We examine the optimal solution property of the model and use it to generate an extended formulation in the higher dimensional space. Then we prove that the constraint matrix for the extended formulation is totally unimodular. We also project it back to a lower dimension space such that we can find valid inequalities that can provide tighter extended formulation of the problem.

5.2 An Extended Formulation for Two Stage SULSB-WW

In this section, we study the optimal solution forms of inventory and backlogging levels for two-stage SUSLB-WW and generate a reformulation which can describe the integral polyhedra of the problem in the higher dimension space.

First, we define Wanger-Whitin costs for two-stage SULSB. We substite $x_i = d_i + s_i - \ell_i - s_{i^-} + \ell_{i^-}$ to eliminate decision variable x_i in two-stage SULSB. Then, we get a reformulation of two-stage SULSB in (s, ℓ, z) space as follows:

$$
\begin{aligned}
& \sum_{i=1}^{p}(\alpha_i x_i + \beta_i' z_i + h_i' s_i + b_i' \ell_i) + \sum_{w=1}^{W} \rho_w \left[\sum_{i \in \mathcal{P}_w, t(i) \geq t(p)+1} (\alpha_i x_i + \beta_i' z_i + h_i' s_i + b_i' \ell_i) \right] \\
= \; & \sum_{i=1}^{p}[\alpha_i(d_i + s_i - \ell_i - s_{i-1} + \ell_{i-1}) + \beta_i' z_i + h_i' s_i + b_i' \ell_i] \\
& + \sum_{w=1}^{W} \rho_w \left[\sum_{i \in \mathcal{P}_w, t(i) \geq t(p)+1} \alpha_i(d_i + s_i - \ell_i - s_{i^-} + \ell_{i^-}) + \beta_i' z_i + h_i' s_i + b_i' \ell_i \right] \\
= \; & \sum_{i=1}^{p-1}[(\alpha_i + h_i - \alpha_{i+1}) s_i + (b_i' - \alpha_i + \alpha_{i+1}) \ell_i + \beta_i' z_i] + \sum_{i=1}^{p} \alpha_i d_i \\
& + (\alpha_p + h_p' - \sum_{w=1}^{W} \rho_w \alpha_{q_w}) s_p + (b_p' - \alpha_p + \sum_{w=1}^{W} \rho_w \alpha_{q_w}) \ell_p + \beta_p' z_p \\
& + \sum_{w=1}^{W} \sum_{j \in \mathcal{P}_w, t(j) \geq t(p)+1} \rho_w (h_j' + \alpha_j - \alpha_{\mathcal{C}(j)}) s_j + \sum_{w=1}^{W} \sum_{j \in \mathcal{P}_w, t(j) \geq t(p)+1} \rho_w (b_j' - \alpha_j + \alpha_{\mathcal{C}(j)}) \ell_j
\end{aligned}
$$

$$
\begin{aligned}
&+\sum_{w=1}^{W}\sum_{j\in\mathcal{P}_w,t(j)\geq t(p)+1}\rho_w\beta'_j z_j+\sum_{w=1}^{W}\sum_{j\in\mathcal{P}_w,t(j)\geq t(p)+1}\rho_w\alpha_j d_j\\
=\ &\sum_{i\in\mathcal{V}}(h_i s_i+b_i\ell_i+\beta_i z_i)+(\text{a constant}),
\end{aligned}
$$

where

$$
h_i=\begin{cases}\alpha_i+h'_i-\alpha_{i+1}, & 1\leq t(i)\leq t(p)-1\\ \alpha_p+h'_p-\sum_{w=1}^{W}\rho_w\alpha_{q_w}, & t(i)=t(p)\\ \rho_w(h'_i+\alpha_i-\alpha_{\mathcal{C}(i)}), & t(i)\geq t(p)+1.\end{cases}
\qquad
b_i=\begin{cases}b'_i-\alpha_i+\alpha_{i+1}, & 1\leq t(i)\leq t(p)-1\\ b'_p-\alpha_p+\sum_{w=1}^{W}\rho_w\alpha_{q_w}, & t(i)=t(p)\\ \rho_w(b'_i-\alpha_i+\alpha_{\mathcal{C}(i)}), & t(i)\geq t(p)+1.\end{cases}
$$

and $\beta_i=\beta'_i$, if $1\leq t(i)\leq t(p)$; otherwise, $\beta_i=\rho_w\beta'_i$. We let $\alpha_{\mathcal{C}(i)}=0$ if $i\in\mathcal{L}$, where $\mathcal{C}(i)$ represents the set of children of node i and $\mathcal{L}$ represents the set of leaf nodes.

Definition 2. *A two-stage stochastic lot-sizing problem with backlogging is said to have Wagner-Whitin costs if*

$$
h_i\geq 0,\ \ b_i\geq 0,\ \ \textit{and}\ \beta_i\geq\sum_{j\in\mathcal{C}(i)}\beta_j \tag{5–1}
$$

for all $i\in\mathcal{V}$.

For the deterministic version, i.e., ULSB with Wagner-Whitin costs, the optimal solution forms of inventory and backlooging levels are studied by Pochet and Wolsey (1994). For two-stage SULSB, we consider the optimal solution forms of inventory level s_i and backlogging level ℓ_i for each $i\in\mathcal{V}$. In the optimal solution for a two-stage SULSB with Wagner-Whitin costs, the demand for each node i will be satisfied (1) by setting up the production at node i, (2) by inventory left from its parent node, or (3) by backlogging from its children. Before we describe the proposition, we show a 3-period example to demonstrate the optimal solution form as shown in Figure 5-1.

In this example, productions are set up at nodes 1, 3, and 4. Demand at node 1 is satisfied by the production of itself. Demand at node 2 is satisfied by backlogging from node 4. Node 3 covers demands in nodes 3 and 5. Thus the second stage nodes 2 and

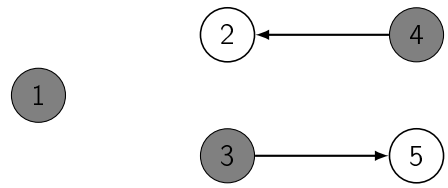


Figure 5-1. The scenario tree for a 3 period SULS with backlogging

3 are backlogged and set up respectively. Meanwhile, the first stage node 1 does not provide inventory for any second stage node.

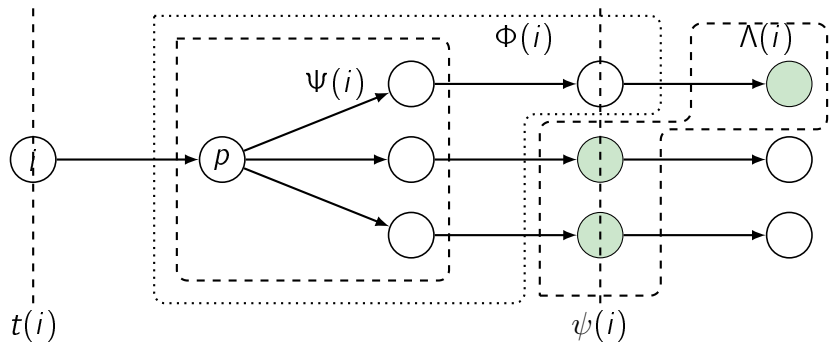


Figure 5-2. The subtree of node i

Now we consider the optimal solution forms of inventory level s_i and backlogging level ℓ_i for two-stage SULSB-WW. First, we introduce binary decision variables f_i and g_i to indicate if node i is stocked or backlogged. If node i is stocked, $f_i = 1$; otherwise, $f_i = 0$. If node i is backlogged, $g_i = 1$; otherwise, $g_i = 0$. Second, extra notation is introduced. As shown in Figure 5-2, let $\psi(i) = \min\{t(j) : z_j = 1 \text{ or } g_j = 1, j \in \mathcal{V}(i) \setminus \{i\}\}$. That is, $\psi(i)$ represents the time period of the earliest descendant of node i which is set up or backlogged. Accordingly, we define node set $\Psi(i) = \{r : r \in \mathcal{V}(i),\ t(i) < t(r) < \psi(i)\}$. Let $\Lambda(i) = \bigcup_{1 \le \omega \le W} \arg\min\{t(k) : k \in \mathcal{P}_T^i(\omega) \setminus \{i\} \text{ and } z_k = 1\}$ and $\Phi(i) = \cup_{j \in \Lambda(i)} \mathcal{P}(i,j) \setminus \{i,j\}$, where $\mathcal{P}_t^i(\omega)$ represents the path from node i to node ℓ which is at time period t and on the branch corresponding to scenario ω. From the observation, it is obvious that the optimal inventory level s_i covers the demands in periods after period $t(i)$ and before period $\psi(i)$. Each descendant of node i in time period $\psi(i)$ sets up or gets backlogging. And each descendant of node i in set $\Lambda(i)$ sets

up. In the following proposition, we provide the closed form of the inventory level s_i and the backlogging level ℓ_i, $i \in \mathcal{V}$.

Proposition 5.1. *For two-stage SULSB-WW,(1) there exists an optimal inventory level of the form:*

$$s_i = \sum_{t=t(i)+1}^{\psi(i)-1} d_t, \qquad i \in \mathcal{V}, \tag{5–2}$$

and an optimal backlogging level of the form:

$$\ell_j = \max_{1 \le \tau \le t(j)} \sum_{t=\tau}^{t(j)} d_t \left[g_{\eta(j,t)} - \sum_{r \in \mathcal{P}(\eta(j,t),j)} z_r \right]^+, \qquad j \in \Phi(i) \tag{5–3}$$

where $\eta(j, t)$ is the ancestor node of node j at time t. That is, $\eta(j, t) = \{k \in \mathcal{P}(j) : t(k) = t\}$.

(2) The optimal solutions of two-stage SULSB-WW satisfy

$$f_i + g_i + z_i = 1, \qquad i \in \mathcal{V} \tag{5–4}$$

$$g_i + z_i \ge g_{i^-}, \qquad i \in \mathcal{V} \setminus \{1\} \tag{5–5}$$

$$f_{i^-} + z_{i^-} \ge f_i, \qquad i \in \mathcal{V} \setminus \{1\} \tag{5–6}$$

Proof. Let $\Lambda = \{i \in \mathcal{V}: z_i = 1\}$. First, for each $i \in \Lambda$, we prove that (5–2) and (5–3) hold by showing (5–2), and

$$\ell_j = \sum_{t=\psi(i)}^{t(j)} d_t \qquad \text{for each } j \in \Phi(i) \setminus \Psi(i), j \in \Phi(i) \setminus \Psi(i) \tag{5–7}$$

and (5–3) hold.

Proof of (5–7). We prove (5–7) under 3 cases.

Case 1: (5–2) does not hold and $\ell_j = \sum_{t=\psi(i)}^{t(j)} d_t$ holds. In this case, s_i^* is either larger or less than $\sum_{t=t(i)+1}^{\psi(i)-1} d_t$.

Case 1.1. If $s_i^* < \sum_{t=t(i)+1}^{\psi(i)-1} d_t$, then there exists at least one node j in which $t(j) = \psi(i) - 1$ whose demand is not satisfied. Thus, s_i^* is not a feasible solution.

Case 1.2. If $s_i^* > \sum_{t=t(i)+1}^{\psi(i)-1} d_t$, then let $\overline{s_i} = \sum_{t=t(i)+1}^{\psi(i)-1} d_t - \varepsilon$, where $0 < \varepsilon \leq s_i^* - \sum_{t=t(i)+1}^{\psi(i)-1} d_t$. It can be observed that $(\overline{s}, \ell^*)$ is also a feasible solution and leads to a non-larger total cost, which is a contradiction. Thus Claim 1 holds.

Case 2: $\ell_j = \sum_{t=\psi(i)}^{t(j)} d_t$ does not hold, but (5–2) holds. We can give the similar proof as in Case 1 to find a contradiction.

Case 3: Neither (5–2) or nor $\ell_j = \sum_{t=\psi(i)}^{t(j)} d_t$ holds. In order to satisfy the demand in each node in $\Phi(i)$, we can construct two feasible solutions for two-stage SULSB-WW.

Let $s_i^1 = s_i^* + \varepsilon$, where ε is a small positive number and node j in $\Lambda(i)$ produces ε less. Then $\ell_j^1 = \ell_j^* - \varepsilon$, $j \in \Phi(i) \setminus \Psi(i)$. The corresponding objective value is

$$F^1 = \sum_{i \in \mathcal{V}} (h_i s_i + b_i \ell_i + \beta_i y_i) + \sum_{j \in \Psi(i)} h_j \varepsilon - \sum_{j \in \Phi(i) \setminus \Psi(i)} b_j \varepsilon.$$

Let $s_i^2 = s_i^* - \varepsilon$, and node j in $\Lambda(i)$ produces ε more. Then $\ell_j^2 = \ell_j^* + \varepsilon$, $j \in \Phi(i) \setminus \Psi(i)$. The corresponding objective value is

$$F^2 = \sum_{i \in \mathcal{V}} (h_i s_i + b_i \ell_i + \beta_i y_i) - \sum_{j \in \Psi(i)} h_j \varepsilon + \sum_{j \in \Phi(i) \setminus \Psi(i)} b_j \varepsilon.$$

If $\sum_{j \in \Psi(i)} h_j < \sum_{j \in \Phi(i) \setminus \Psi(i)} b_j$, then $F^1 < F^*$; F^* is the optimal objective value.

If $\sum_{j \in \Psi(i)} h_j > \sum_{j \in \Phi(i) \setminus \Psi(i)} b_j$, then $F^2 < F^*$; This contradicts with the assumption that F^* is the optimal objective value.

Note here, if $\sum_{j \in \Psi(i)} h_j = \sum_{j \in \Phi(i) \setminus \Psi(i)} b_j$, we increase (or decrease) s^* and decrease (or increase) ℓ^* to match the optimal form. Therefore, (5–7) holds.

Proof of (5–3): According to (5–7), s_i covers demands for nodes in $\Psi(i)$. In order to minimize the objective function, nodes in $\Psi(i)$ do not obtain backlogging. Thus $g_j = 0$, $j \in \Psi(i)$. Therefore, $\ell_j = \max_{\{\tau: 1 \leq \tau \leq t(j)\}} \sum_{t=\tau}^{t(j)} d_t \left[g_{\eta(j,t)} - \sum_{r \in \mathcal{P}(\eta(j,t),j) \setminus \{j\}} z_r \right]^+ = \sum_{t=\psi(i)}^{t(j)} d_t$.

($\ell_j = \max_{\{\tau: 1 \leq \tau \leq t(j)\}} \sum_{t=\tau}^{t(j)} d_t \left[g_{\eta(j,t)} - \sum_{r \in \mathcal{P}(\eta(j,t),j)} z_r \right]^+ = \sum_{t=\psi(i)}^{t(j)} d_t$.)

Therefore, (5–2) and (5–3) hold.

Second, we prove that conditions (5–4), (5–5), and (5–6) hold.

According to the definition of f_i and g_i, $i \in \mathcal{V}$, it is obvious that Conditions (5–5) and (5–6) hold.

Now we prove that Condition (5–4) holds by two cases.

Case 1. If $j \in \Psi(i)$, then the demand in node j is satisfied by inventory and $f_j = 1$. In order to keep the smallest production cost, $f_j + g_j + z_j = 1$.

Case 2. If $j \in \Phi(i) \setminus \Psi(i)$, then $f_j = 0$. and we need to prove $g_j + z_j = 1$. In order to satisfy the demand, $g_j + z_j \geq 1$.

Case 2.1. If $j \in \Lambda(i)$, $z_j = 1$. If $g_i = 1$, then we can let node j produce more to cover the backlogging amount and reduce the objective value. Contradiction! Therefore, $g_j = 0$ and $f_j + g_j + z_j = 1$.

Case 2.2. If $j \in \Phi(i) \setminus \Psi(i)$, $g_j = 1$. According to (5–3), the demand of node j can be covered by backlogging from its children. In order to minimize the objective function, $z_j = 0$.

Therefore, based on cases 1 and 2, $f_j + g_j + z_j = 1, j \in \Phi(i)$. □

Under (5–4), at most one of $g_{\eta(i,k)}$ and $z_{\eta(i,k)}$ equals to 1. Then $g_{\eta(i,k)} - z_{\eta(i,k)} = 1$ if $g_{\eta(i,k)} = 1$; $g_{\eta(i,k)} - z_{\eta(i,k)} \leq 0$ if $g_{\eta(i,k)} = 0$. Thus,

$$\begin{aligned}
\ell_j &= \max_{\tau} \sum_{t=\tau}^{t(j)} d_t \left[g_{\eta(j,t)} - \sum_{r \in \mathcal{P}(\eta(j,t),j) \setminus \{j\}} z_r \right]^+ \\
&= \max_{\tau} \sum_{t=\tau}^{t(j)} d_t \left[g_{\eta(j,t)} - z_{\eta(j,t)} - \sum_{r \in \mathcal{P}(\eta(j,t),j) \setminus \{j\}} z_r \right]^+ \\
&= \max_{\tau} \sum_{t=\tau}^{t(j)} d_t \left[g_{\eta(j,t)} - \sum_{r \in \mathcal{P}(\eta(j,t),j) \setminus \{j, \eta(j,t)\}} z_r \right]^+, \qquad 1 \leq \tau \leq t(j). \qquad (5\text{–}8)
\end{aligned}$$

From (5–4), we have

$$f_i + g_i + z_i = f_{i^-} + g_{i^-} + z_{i^-} = 1. \tag{5–9}$$

Thus one of constraints (5–5) and (5–6) is redundant. That is, if constraint (5–5) holds, then (5–6) must hold from (5–9).

Let binary variable m_t^i represent whether the inventory left from node i covers the demand at time period t, $t \geq t(i) + 1$. If yes, $m_t^i = 1$; otherwise, $m_t^i = 0$. Let binary variable n_t^i represent whether the backlogging at node i covers the demand at time period t, $t \leq t(i)$. If yes, $n_t^i = 1$; otherwise $n_t^i = 0$. Finally we let $\xi(i, t, w)$ represent the node which is a descendant of node i at time period t and at the branch corresponding to scenario w.

Now we introduce three types of inequalities corresponding to the optimal inventory level for node i on the scenario tree.

1. Path I inequality: this type of inequality is for the second stage nodes. For a given node i on the second stage, w is determined and

$$m_t^i \geq f_{\xi(i,t,w)} - \sum_{j \in \mathcal{P}_t^i(w) \setminus \{i, \xi(i,t,w)\}} z_j, \qquad t(i) \geq t(p) + 1, \ t \geq t(i) + 1. \ i > p. \tag{5–10}$$

That is, node i covers demands along the branch it belongs to until next backlogging or production happens.

2. Path II inequality: this type of inequality is for the first stage nodes (except node p). For a given node i on the first stage, node $i + 1$ is its child node and

$$m_t^i \geq f_{i+1}, \quad 1 \leq t(i) \leq t(p) - 1, t = t(i) + 1, \tag{5–11}$$

$$m_t^i \geq m_t^{i+1} - z_{i+1}, \quad 1 \leq t(i) \leq t(p) - 1, t \geq t(i) + 2. \tag{5–12}$$

Inequality (5–11) indicates that if the demand of child node $i + 1$ is satisfied by inventory, then the inventory left from node i covers the demand of node $i + 1$. Inequality (5–12) indicates that if node $i + 1$ is not set up and the inventory left from

node $i+1$ covers the demand up to time t, $t \geq t(i)+2$, then the inventory left from node i also covers the demand up to time t.

3. Connection inequality: this type of inequality is for the branching node p,

$$m_t^p \geq f_{q_w}, \quad q_w \in \mathcal{C}(p),\, t = t(p)+1, \tag{5–13}$$

$$m_t^p \geq m_t^{q_w} - z_{q_w}, \quad q_w \in \mathcal{C}(p),\, t \geq t(p)+2. \tag{5–14}$$

Inequalities (5–13) and (5–14) are similar to (5–11) and (5–12). The inventory left from node p will cover demands from period $t(p)+1$ to $t-1$ unless there is a setup or a backlogging, before or at time period t along each scenario path.

With the information of m_t^i, the inventory level left from each node i in the scenario tree is as follows:

$$s_i = \sum_{t=t(i)+1}^{T} d_t m_t^i, \qquad i \in \mathcal{V} \tag{5–15}$$

Because for a given node i, its ancestor at time period t, $\eta(i,t)$, is unique, we have the following two inequalities hold for each node $i \in \mathcal{V}$ based on (5–8).

$$n_t^i \geq g_{\eta(i,t)} - \sum_{j \in \mathcal{P}(\eta(i,t),i)\setminus\{\eta(i,t)\}} z_j, \qquad i \in \mathcal{V} \text{ and } t \leq t(i). \tag{5–16}$$

$$\ell_i = \sum_{t=1}^{t(i)} d_t n_t^i, \qquad i \in \mathcal{V}. \tag{5–17}$$

Constraints (5–4), (5–5), (5–10) - (5–17) guarantee the feasibilities of the reformulation for two-stage SULSB-WW problem since the demand for each time period is covered. Now, we show that constraints provide the extended formulation for two stage SULSB-WW.

Proposition 5.2. *Constraints (5–4), (5–5), and (5–10) to (5–17) provide the extended formulation for the two-stage SULSB-WW problem.*

Proof. Because constraints (5–15) and (5–17) can be directly transferred to the objective function. We prove this proposition by showing the constraint matrix for constraints (5–4), (5–5), (5–10) to (5–14), and (5–16) is a totally unimodular.

To prove the constraint matrix for constraints (5–4), (5–5), (5–10) to (5–14), and (5–16) is totally unimodular, we order variable f_i, g_i, and z_i with loop i ranging from 1 to $|\mathcal{V}|$. Variable m_t^i is ordered with an outer loop i ranging from 1 to $|\mathcal{V}|$ and an inner loop t ranging from $t(i)+1$ to T. n_t^i is ordered with an outer loop i ranging from 1 to $|\mathcal{V}|$ and an inner loop t ranging from 1 to $t(i)$. Table 5-1 shows the constraints matrix corresponding to Figure 5-1.

As the the submatrix corresponding to variable n_t^i is an identity matrix for $i \in \mathcal{V}$ and $t \leq t(i)$, we do not need to consider n_t^i in our construction. As the submatrix corresponding to variable m_t^i is an identity matrix for $t(i) \geq t(p)+1$, $i \in \mathcal{V}$, and $t \geq t(i)+2$, we only need to consider m_t^i, $t(i) \leq t(p)$, $t \geq t(i)+2$ in our construction. That is, we only consider the associated constraint submatrix for the rest variables, denoted as A.

We show that for any column subset J of matrix A, there exists partition J_1 and J_2 of J such that

$$\left| \sum_{j \in J_1} a_{ij} - \sum_{j \in J_2} a_{ij} \right| \leq 1 \tag{5–18}$$

for all i. We do the partition of variables f, g, z, m in J starting from branching node p and then extend it to both direction for nodes after p and before p respectively.

First, we define $M(i)$ as the closest ancestor of node i such that $z_{M(i)} \in J$ and $m(i)$ as the closest descendant of node i such that $z_{m(i)} \in J$.

In the following steps 1 and 5, we allocate the decision variables m and z:

Step 1. Allocate m_t^p to J_1, z_p to J_1, and z_{q_w} to J_2, where $t \geq t(p)+1$, $q_w \in \mathcal{C}(p)$.

Step 2. Allocate $m_t^{q_w}$, $t \geq t(p)+2$ to the same set as z_{q_w} (if $z_{q_w} \in J$), or to the same set as m_t^p (if $z_{q_w} \notin J$ and $m_t^p \in J$), or to J_1 (if $z_{q_w} \notin J$ and $m_t^p \notin J$).

Step 3. Allocate z_i, $t(i) \geq t(p)+2$, to the opposite set of $z_{M(i)}$, if $M(i)$ exists and $t(M(i)) \geq t(p)$. Otherwise, allocate z_i to the opposite set of m_t^p if $m_t^p \in J$. If $m_t^p \notin J$, then allocate z_i to J_2.

Step 4. Allocate z_i, $1 \leq i \leq q-1$, to the opposite set of $z_{D(i)}$, if $m(i)$ exists and $t(m(i)) \leq t(p)$. Otherwise, allocate z_i to J_1.

Step 5. Allocate m_t^i, $1 \leq i \leq q-1$, to the opposite set as $z_{m(i)}$, if there exists $z_{m(i)}$ as shown in Step 4 in J. Otherwise, allocate m_t^i to J_1.

In the following Steps 6 and 7, we allocate the decision variables f and g:

Step 6. Allocate f_i to the same set of $z_{M(i)}$, if $z_{M(i)} \in J$; allocate f_i to the opposite set of z_i, if $z_{M(i)} \notin J$, $z_i \in J$; allocate f_i to the opposite set of $z_{m(i)}$, if $z_{m(i)} \in J$, $z_{M(i)}, z_i \notin J$; allocate f_i to J_1, if $z_{m(i)}, z_{M(i)}, z_i \notin J$.

Step 7. Allocate g_i to the same set of $z_{m(i)}$, if $z_{m(i)} \in J$; allocate g_i to the opposite set of z_i, if $z_{m(i)} \notin J$, $z_i \in J$; allocate g_i to the opposite set of $z_{M(i)}$, if $z_{m(i)}, z_i \notin J$, $z_{M(i)} \in J$; allocate g_i to J_2, if $z_{m(i)}, z_i, z_{M(i)} \notin J$.

Following the above partition steps, we observe the following two properties.

Claim 1. If $\{z_i, z_{M(i)}\} \subseteq J$, z_i goes to the opposite set of $z_{M(i)}$ for all $i \in \mathcal{V} \setminus \{1\}$.

Proof of Claim 1. If $1 \leq t(M(i)) \leq t(i) \leq t(p)$, because the closest descendant of $M(i)$ is i and $t(i) \leq t(p)$, $m(M(i)) = i$, $z_{M(i)}$ goes to the opposite set of z_i based on Step 4.

If $1 \leq t(M(i)) < t(p) < t(i) \leq T$, $z_{M(i)}$ goes to J_1 based on Step 4 and z_i goes to J_2 based on Step 3.

If $t(p) = t(M(i)) < t(i) \leq T$, $z_{M(i)}$ goes to J_1 based on Step 1 (i.e., $M(i) = p$), z_i goes to J_2 based on Step 2 (if $i = q_w$) or Step 3 (if $t(i) \geq t(p) + 2$). Thus, z_i goes to the opposite set of $z_{M(i)}$.

If $t(p) + 1 \leq t(M(i)) < t(i) \leq T$, z_i goes to the opposite set of $z_{M(i)}$ based on Step 3.

Therefore, Claim 1 holds. □

Claim 2. If $j(w_{k_1})$ and $j(w_{k_2})$ be the first second-stage nodes corresponding to scenarios w_{k_1} and w_{k_2} such that $z_{j(w_{k_1})}, z_{j(w_{k_2})} \in J$, then $z_{j(w_{k_1})}$ and $z_{j(w_{k_2})}$ go to the same set.

Proof of Claim 2. For any first second-stage node $j(w)$, if $j(w) = q_w$, $z_{j(w)}$ goes to J_2 based on Step 1, otherwise $j(w)$ goes to J_2 based on Step 3 due to the fact that $j(w)$ is the first second-stage node in J and on the branch corresponding to w and $t(p) \leq t(j(w)) \leq T$. Thus, the claim holds. □

Now we verify that (5–18) holds for constraints (5–4), (5–5), (5–10) to (5–14), and (5–16) under the above partition. At first, corresponding to each row, if J contains at most one decision variable in A, then it is obvious that (5–18) holds. In the following, we consider the case that J contains at least two decision variables in A.

1. For constraint (5–4), we discuss the following 4 cases.

1-1. $\{f_i, g_i\} \subseteq J$ and $z_i \notin J$

1-1-1. If $z_{m(i)}, z_{M(i)} \in J$ both $M(i)$ and $m(i)$ exist, f_i goes to the same set of $z_{M(i)}$ based on Step 6 ; g_i goes to the same set of $z_{m(i)}$ based on Step 7. Because $M(m(i)) = M(i)$ due to $z_i \notin J$, $z_{m(i)}$ goes to the opposite set of $z_{M(i)}$ based on Claim 1. Thus, f_i goes to the opposite set of g_i. Then, (5–18) holds.

1-1-2. If $z_{M(i)} \in J$, $z_{m(i)} \notin J$, $M(i)$ exists, but $m(i)$ does not exist, f_i goes to the same set of $z_{M(i)}$ based on Step 6; g_i goes to the opposite set of $z_{M(i)}$ based on Step 7. Thus, f_i goes to the opposite set of g_i.

1-1-3. If $z_{m(i)} \in J$, $z_{M(i)} \notin J$, $m(i)$ exists, but $M(i)$ does not exist, g_i goes to the same set of $z_{m(i)}$ based on Step 7; f_i goes to the opposite set of $z_{m(i)}$ based on Step 6. Thus, f_i goes to the opposite set of g_i.

1-1-4. If $z_{m(i)},\ z_{M(i)} \notin J$, both $m(i)$ and $M(i)$ do not exist, f_i and g_i go to J_1 and J_2 based on Steps 6 and 7 respectively.

1-2. $\{g_i, z_i\} \subseteq J$ and $f_i \notin J$

1-2-1. If $z_{m(i)} \in J$, m(i) exists, g_i goes to the same set of $z_{m(i)}$ based on Step 7. Because $M(m(i)) = i$, $z_{m(i)}$ goes to the opposite set of z_i based on Claim 1. Thus, g_i goes to the opposite set of z_i. Then, (5–18) holds.

1-2-2. If $z_{m(i)} \notin J$, m(i) does not exist, g_i goes to the opposite set of z_i based on Step 7. Thus, (5–18) holds.

1-3. $\{f_i, z_i\} \in J$ and $g_i \notin J$

1-3-1. If $z_{M(i)} \in J$, M(i) exists, f_i goes to the same set of $z_{M(i)}$ based on Step 6; z_i goes to the opposite set of $z_{M(i)}$ based on Claim 1. Then, f_i goes to the opposite set of z_i. Thus, (5–18) holds.

1-3-2. If $z_{M(i)} \notin J$, M(i) does not exist, f_i goes to the opposite set of z_i based on Step 6. Thus, (5–18) holds.

1-4. $\{f_i, g_i, z_i\} \in J$, this conclusion directly follows 1-3, because f_i goes to the opposite set of z_i. Then (5–18) holds no matter where g_i goes.

2. For constraint (5–5), we discuss the following 4 cases.

2-1. If $\{g_i, z_i\} \in J$ and $g_{i^-} \notin J$ this condition is the same as 1-2.

2-2. If $\{g_{i^-}, z_i\} \in J$ and $g_i \notin J$ g_{i^-} goes to the same set as z_i due to $z_i = z_{m(i^-)}$ based on Step 7. Thus, (5–18) holds.

2-3. If $\{g_i, g_{i^-}\} \in J$ and $z_i \notin J$, we first have $m(i) = m(i^-)$. If (a) $m(i) = m(i^-)$ exists, then g_i and g_{i^-} go to the same set as $z_{m(i)}$ based on Step 7. If (b) $m(i) = m(i^-)$ does not exist and $z_{i^-} \in J$, then g_{i^-} goes to the opposite set of z_{i^-} based on Step 7. Also,, because $z_{i^-} \in J$, we have $z_{M(i)} = z_{i^-}$. Then g_i goes to the opposite set of z_{i^-} based on Step 7. Thus g_i and g_{i^-} go to the same set. If (c) $m(i) = m(i^-)$ does not exist, $z_{i^-} \notin J$ and $M(i^-)$ exists, then $z_{M(i)} = z_{M(i^-)}$. Note that $z_i \notin J$, we have g_i and g_{i^-} go to the opposite set of $z_{M(i)} = z_{M(i^-)}$ based on Step 7. Thus, g_i and g_{i^-} go to the same set. If (d) $m(i) = m(i^-)$, $M(i^-)$ do not exist, $z_{i^-} \notin J$, then both g_i and g_{i^-} go to J_2 based on Step 7.

2-4. If $\{g_i, g_{i^-}, z_i\} \in J$, this conclusion directly follows 2-2, g_{i^-} and z_i go to the same set. Then (5–18) holds no matter where g_i goes.

3. For constraint (5–10), we consider 2 cases:

3-1. $t(i) \geq t(p) + 2$. For this case, first, we do not need to consider m_t^i based on the identity matrix argument at the beginning of the proof. Then, based on Step 6, $f_{\xi(i,t,w)}$ goes to the same set of $z_{M(\xi(i,t,w))}$, where $z_{M(\xi(i,t,w))}$ is the largest-index node in J and path $p_t^i(w)$. Note here, here must exist at least one such $M(\xi(i, t, w))$ based on the assumption that we have at least two elements in J for each constraint. Based on Step 4, z_j alternatively goes to J_1 and J_2 based on Step 4, where $t(p) + 2 \leq t(j) \leq t$. Thus, (5–18) holds.

3-2. $t(i) = t(p) + 1$. For this case, $i = q_w$, for some $1 \leq w \leq W$. We discuss the following two cases.

3-2-1. If $m_t^{q_w} \notin J$, this argument is the same as 3-1.

3-2-2. If $m_t^{q_w} \in J$, under this condition, we discuss $f_{\xi(q_w,t,w)} \notin J$ and $f_{\xi(q_w,t,w)} \in J$ respectively.

3-2-2-1. $f_{\xi(q_w,t,w)} \notin J$. Under this case, if $z_{q_w} \in J$, $m_t^{q_w}$ and $z_{m(q_w)}$ go to the same set and the opposite set of z_{q_w} based on Steps 2 and 3 respectively; if $z_{q_w} \notin J$, $m_t^{q_w}$ goes to the same set as m_t^p (i.e., J_1) if $m_t^p \in J$ or J_1 if $m_t^p \notin J$ based on Step 2 and similarly $z_{m(q_w)}$ goes to J_2 based on Step 3. Thus, $m_t^{q_w}$ and $z_{m(q_w)}$ go to the opposite sets. Besides these, $z_j \in J$ alternatively goes to J_1 and J_2 where $t(p) + 2 \leq t(j) \leq t$ based on Step 3. Thus, (5–18) holds.

3-2-2-2. $f_{\xi(q_w,t,w)} \in J$. Under this case, we discuss two cases depending on if there exists a node $j \in \mathcal{P}_t^i(w) \setminus \{i, \xi(i, t, w)\}$ such that $z_j \in J$.

3-2-2-2-1. If no such node j exists, then $\{f_{\xi(q_w,t,w)}, m_t^{q_w}\} \in J$, based on our assumption that at least two elements in each constraints in matrix A. If $z_{q_w} \in J$, $M(\xi(q_w, t, w)) = q_w$. Based on Steps 6 and 2, $f_{\xi(q_w,t,w)}$ and $m_t^{q_w}$ go to the same set of z_{q_w} . If $z_{q_w} \notin J$, based on Step 2, $m_t^{q_w}$ goes to J_1. In the following, we prove that $f_{\xi(q_w,t,w)}$ goes to J_1 in this case. Based on Step 6,

$f_{\xi(q_w,t,w)}$ goes to (a) the same set of $z_{M(\xi(q_w,t,w))}$ (if $z_{M(\xi(q_w,t,w))} \in J$ if $M(\xi(q_w, t, w))$ exists). For this case, $z_{M(\xi(q_w,t,w))} \in J$ $M(\xi(q_w, t, w))$ exists and $z_{q_w} \notin J$. Based on Step 4, $z_{M(\xi(q_w,t,w))}$ goes to J_1, because $1 \leq t(M(\xi(q_w, t, w))) \leq t(p)$. (b) the opposite set of $z_{\xi(q_w,t,w)}$ (if $z_{M(\xi(q_w,t,w))} \notin J$, $M(\xi(q_w, t, w))$ does not exist, $z_{\xi(q_w,t,w)} \in J$). For this case, $z_{\xi(q_w,t,w)} \in J$ and $z_{M(\xi(q_w,t,w))}, z_{q_w} \notin J$, $z_{q_w} \notin J$ and $M(\xi(q_w, t, w))$ does not exist. Then, $z_{\xi(q_w,t,w)}$ goes to J_2 based on Step 3 and according $f_{\xi(q_w,t,w)}$ goes to J_1. (c) the opposite set of $z_{m(\xi(q_w,t,w))}$ (if $z_{M(\xi(q_w,t,w))}, z_{\xi(q_w,t,w)} \notin J$, $z_{m(\xi(q_w,t,w))}) \in J$, $m(\xi(q_w, t, w))$ exists, but $M(\xi(q_w, t, w))$ does not exist, $z_{\xi(q_w,t,w)} \notin J$). For this case, $z_{m(\xi(q_w,t,w))} \in J$, $z_{\xi(q_w,t,w)}, z_{M(\xi(q_w,t,w))}, z_{q_w} \notin J$. Then, $z_{m(\xi(q_w,t,w))}$ goes to J_2 based on Step 3. Thus, accordingly $f_{\xi(q_w,t,w)}$ goes to J_1. Then $f_{\xi(q_w,t,w)}$ and $m_t^{q_w}$ go to the same set. (d) J_1 (if $z_{M(\xi(q_w,t,w))}, z_{\xi(q_w,t,w)}, z_{m(\xi(q_w,t,w))} \notin J$ $M(\xi(q_w, t, w))$ and $m(\xi(q_w, t, w))$ do not exist and $z_{\xi(q_w,t,w)} \notin J$).

3-2-2-2-2. If there exists a such node j, then $f_{\xi(q_w,t,w)}$ goes to the same set of $z_{M(\xi(q_w,t,w))}$ based on Step 6. If $z_{q_w} \in J$, $m_t^{q_w}$ goes to the same set of z_{q_w} based on Step 2, and $z_{m(q_w)}$ goes to the opposite set of z_{q_w} based on Step 3. If $z_{q_w} \notin J$, $m_t^{q_w}$ goes to J_1 based on Step 2 and $z_{m(q_w)}$ goes to J_2 based on Step 3. Thus, for both $z_{q_w} \in J$ and $z_{q_w} \notin J$, we have $m_t^{q_w}$ goes to the opposite set of $z_{m(q_w)}$. Besides these, z_j alternatively goes to J_1 and J_2, $j \in \mathcal{P}_t^i(w) \setminus \{i, \xi(i, t, w)\}$. Thus, (5–18) holds.

4. For constraint (5–11), we only need to consider the case if $\{m_t^i, f_{i+1}\} \in J$.

4-1. If $M(i+1)$ exists, f_{i+1} goes to the same set as $z_{M(i+1)}$ based on Step 6. If $m(i)$ exists and $t(m(i)) \leq t(p)$, m_t^i goes to the opposite set of $z_{m(i)}$ based on Step 5. Because $m(i) = m(M(i+1))$, $z_{M(i+1)}$ and $z_{m(i)}$ go to the opposite set based

on Step 4. Thus f_{i+1} and m_t^i go to the same set. If $m(i)$ does not exist, or $m(i)$ exists and $t(m(i)) > t(p)$, then $m(M(i+1))$ does not exists. Thus, $z_{M(i+1)}$ and m_t^i go to J_1 based on Steps 4 and 5. Thus m_t^i and f_{i+1} go to the same set. Therefore, (5–18) holds.

4-2. If $M(i+1)$ does not exist and $z_{i+1} \in J$, then $m(i) = i+1$, Thus, m_t^i and f_{i+1} go to the opposite set of z_{i+1} based on Steps 5 and 6. Thus, (5–18) holds.

4-3. If $M(i+1)$ does not exist, $z_{i+1} \notin J$, and $m(i+1)$ exists, then, $m(i) = m(i+1)$. If $1 \leq t(m(i+1)) \leq t(p)$, m_t^i and f_{i+1} go to the opposite set of $z_{m(i+1)}$ based on Steps 5 and 6; otherwise, if $t(m(i+1)) > t(p)$, m_t^i goes to J_1 and f_{i+1} goes to the opposite set of $z_{m(i+1)}$. Since $z_{m(i+1)}$ goes to J_2 based on Step 1 (if $t(m(i+1)) = t(p)+1$) or Step 3, under this case, $M(m(i+1))$ does not exist. f_{i+1} and m_t^i go to the same set. Then (5–18) holds.

4-4. If neither $M(i+1)$ nor $m(i+1)$ exists and $z_{i+1} \notin J$, m_t^i and f_{i+1} go J_1 based on Steps 5 and 6. Then, (5–18) holds.

5. For constraint (5–12), we discuss the following four cases:

5-1. $\{m_t^i, z_{i+1}\} \subseteq J$, and $m_t^{i+1} \notin J$. Based on Step 5, m_t^i goes to the opposite set of z_{i+1} since z_{i+1} is the closest descendant of node i. Thus, (5–18) holds.

5-2. $\{m_t^{i+1}, z_{i+1}\} \subseteq J$, and $m_t^i \notin J$. Under this case, m_t^{i+1} and z_{i+1} will go to the same set, due to both z_{i+1} and m_t^{i+1} are in the opposite set of $z_{m(i)}$ ($z_{m(i+1)}$), if $m(i+1)$ exists and $t(m(i+1)) \leq t(p)$ or in J_1 otherwise (otherwise, in J_1), based on Steps 4 and 5.

5-3. $\{m_t^i, m_t^{i+1}\} \subseteq J$, and $z_{i+1} \notin J$. Because $z_{i+1} \notin J$, m_t^i and m_t^{i+1} are in the same set based on Step 5.

5-4. $\{m_t^i, m_t^{i+1}, z_{i+1}\} \subseteq J$. The conclusion follows from the fact that based on Step 5. m_t^i goes to the opposite set of z_{i+1}, since z_{i+1} is the closest descendant of node i. Then we have (5–18) holds no matter where m_t^{i+1} goes.

6. For constraint (5–13), we only need to consider the case in which $\{f_{q_w}, m_t^p\} \subseteq J$. Based on Step 1, m_t^p goes to J_1. In the following, we show that f_{q_w} goes to J_1 for the following four cases.

6-1 If $M(q_w)$ exists, then $z_{M(q_w)}$ goes to J_1 based on Step 1 if $M(q_w) = p$, or Step 4 because $(m(M(q_w))) > t(p)$. Thus, based on Step 6, f_{q_w} goes to the same set of $z_{M(q_w)}$, which is J_1. f_{q_w} and m_t^p go to the same set. Thus, (5–18) holds.

6-2 If $z_{q_w} \in J$, $M(q_w)$ does not exist, then z_{q_w} goes to J_2 based on Step 1. Based on Step 6, f_{q_w} goes to the opposite set of z_{q_w}, which is J_1.

6-3 If $m(q_w)$ exists, $z_{q_w} \notin J$ and $M(q_w)$ does not exist, then $t(m(q_w)) \geq t(p) + 2$ and $z_{m(q_w)}$ goes to the opposite set of m_t^p based on Step 3. Based on Step 6, we have f_{q_w} goes to the opposite set of $z_{m(q_w)}$.

6-4 If $m(q_w)$ and $M(q_w)$ do not exist, $z_{q_w} \notin J$, f_{q_w} goes J_1based on Step 6. Thus, f_{q_w} and m_t^p go to the same set.

7. For constraint (5–14), we need to consider the following four cases.

7-1. $\{m_t^p, z_{q_w}\} \subseteq J$, and $m_t^{q_w} \notin J$. Based on Step 1, m_t^p and z_{q_w} go to J_1 and J_2 respectively. Thus (5–18) holds.

7-2. $\{m_t^{q_w}, z_{q_w}\} \subseteq J$, and $m_t^p \notin J$. $m_t^{q_w}$ and z_{q_w} go to the same set based on Step 2.

7-3. $\{m_t^p, m_t^{q_w}\} \subseteq J$, $z_{q_w} \notin J$. Under this case, m_t^p and $m_t^{q_w}$ go to the same set based on step 2.

7-4. $\{m_t^p, m_t^{q_w}, z_{q_w}\} \subseteq J$. Under this case, m_t^p and z_{q_w} go to J_1 and J_2 respectively based on Step 1. Then (5–18) holds no matter where $m_t^{q_w}$ goes.

8. For constraint (5–16), note here, we do not need to consider n_t^i based on the identity matrix argument at the beginning of the proof. Based on Step 7, $g_{\eta(i,t)}$ goes to the same set of $z_{m(\eta(i,t))}$. If $t(m(\eta(i,t))) \leq t(p)$ or $t(\eta(i,t)) \geq t(p) + 1$, then $\eta(i,t)$ and $m(\eta(i,t))$ are one to one corresponding. Otherwise, all $z_{m(\eta(i,t))}$ in different scenarios go to the same set based on Claim 2 because $t(m(\eta(i,t))) \geq t(p) + 1$.

Besides these, z_j alternatively goes to J_1 and J_2 based on Claim 1. Thus, (5–18) holds.

Therefore, the desired property (5–18) holds for constraints (5–4), (5–5), (5–10) to (5–14), and (5–16), and the corresponding constraint matrix is totally unimodular. Thus, the extended formulation provides an integral solution for two-stage SULSB-WW. □

Now we study the integral polyhedra in the (f, g, z, s, ℓ) space to generate a tighter extended formulation for the two-stage SULSB-WW. First, we define

$$
\begin{aligned}
Q_M = \{&(f, g, z, s, \ell) : (f, g, z, s, \ell) \text{ satisfies} \\
&s_i \geq \sum_{t=t(i)+1}^{\tau} d_t\Big(f_{\xi(i,t,w_t)} - \sum_{j \in \mathcal{P}_t^i(w_t) \setminus \{i, \xi(i,t,w_t)\}} z_j\Big), \quad w_t \in W(i), t(i)+1 \leq \tau \leq T, i \in \mathcal{V}, \qquad (5\text{–}19) \\
&\ell_i \geq \sum_{t=\tau}^{t(i)} d_t\Big(g_{\eta(i,t)} - \sum_{j \in \mathcal{P}(\eta(i,t),i) \setminus \{\eta(i,t)\}} z_j\Big), \quad 1 \leq \tau \leq t(i), i \in \mathcal{V}, \qquad (5\text{–}20) \\
&(5\text{–}4) \text{ and } (5\text{–}5), \\
&0 \leq f_i, g_i, z_i \leq 1, s_i, \ell_i \geq 0, \ i \in \mathcal{V}\}.
\end{aligned}
$$

We prove that Q_M is an integral polyhedra for the two-stage SULSB-WW in the (f, g, z, s, ℓ) space, by showing that it is a projection of Q_H in the (f, g, z, s, ℓ) space, where Q_H records the polyhedra of the two-stage SULSB-WW in the (f, g, z, m, n, s, ℓ) space, i.e.

$$
\begin{aligned}
Q_H = \{&(f, g, z, m, n, s, \ell) : (f, g, z, m, n, s, \ell) \text{ satisfies} \\
&(5\text{–}4), (5\text{–}5), (5\text{–}10) \text{to } (5\text{–}17), \\
&0 \leq f_i, g_i, z_i, m_t^i, n_{t'}^i \leq 1, s_i, \ell_i \geq 0, \quad t(i)+1 \leq t \leq T, 1 \leq t' \leq t(i), i \in \mathcal{V}\}.
\end{aligned}
$$

Proposition 5.3. $Proj_{(f,g,z,s,\ell)} Q_H = Q_M$.

Proof. We prove this proposition by showing the following two claims.

Claim 1. All inequalities in Q_M are valid for Q_H.

Claim 2. For an arbitrary extreme point $(f, g, z, s, \ell) \in Q_M$, there exists a $(f, g, z, m, n, s, \ell) \in Q_H$.

Proof of Claim 1. We prove Claim 1 by showing that

$$s_i \geq \sum_{t=t(i)+1}^{T} d_t[f_{\xi(i,t,w_t)} - \sum_{j \in \mathcal{P}_t^i(w_t) \setminus \{i, \xi(i,t,w_t)\}} z_j]^+, \qquad w_t \in W(i), \tag{5–21}$$

$$\ell_i \geq \sum_{t=1}^{t(i)} d_t[g_{\eta(i,t)} - \sum_{j \in \mathcal{P}(\eta(i,t),i) \setminus \{\eta(i,t)\}} z_j]^+ \tag{5–22}$$

are valid for Q_H, because if (5–21) and (5–22) hold, then

$$\begin{aligned}
s_i &\geq \sum_{t=t(i)+1}^{T} d_t[f_{\xi(i,t,w_t)} - \sum_{j \in \mathcal{P}_t^i(w_t) \setminus \{i, \xi(i,t,w_t)\}} z_j]^+ \\
&\geq \sum_{t=t(i)+1}^{\tau} d_t(f_{\xi(i,t,w_t)} - \sum_{j \in \mathcal{P}_t^i(w_t) \setminus \{i, \xi(i,t,w_t)\}} z_j), \qquad t(i)+1 \leq \tau \leq T; \\
\ell_i &\geq \sum_{t=1}^{t(i)} d_t[g_{\eta(i,t)} - \sum_{j \in \mathcal{P}(\eta(i,t),i) \setminus \{\eta(i,t)\}} z_j]^+ \\
&\geq \sum_{t=\tau}^{t(i)} d_t(g_{\eta(i,t)} - \sum_{j \in \mathcal{P}(\eta(i,t),i) \setminus \{\eta(i,t)\}} z_j), \qquad 1 \leq \tau \leq t(i).
\end{aligned}$$

Based on (5–16), (5–17) and the nonnegativity of n_t^i, we have

$$\begin{aligned}
\ell_i &= \sum_{t=1}^{t(i)} d_t n_t^i \geq d_t \sum_{t=1}^{t(i)} \max\{0, g_{\eta(i,t)} - \sum_{j \in \mathcal{P}(\eta(i,t),i) \setminus \{\eta(i,t)\}} z_j\} \\
&= \sum_{t=1}^{t(i)} d_t[g_{\eta(i,t)} - \sum_{j \in \mathcal{P}(\eta(i,t),i) \setminus \{\eta(i,t)\}} z_j]^+
\end{aligned}$$

Thus, (5–22) holds.

Now we prove (5–21) by 2 conditions. (1) $t(i) \geq t(p) + 1$; (2) $t(i) \leq t(p)$.

For Condition 1, $t(i) \geq t(p)+1$, based on (5–10), (5–15), and the nonnegativity of m_t^i, we have

$$s_i = \sum_{t=t(i)+1}^{T} d_t m_t^i \geq \sum_{t=t(i)+1}^{T} d_t \max\left\{0,\, f_{\xi(i,t,w_t)} - \sum_{j\in\mathcal{P}_t^i(w_t)\setminus\{i,\xi(i,t,w_t)\}} z_j\right\}$$
$$= \sum_{t=t(i)+1}^{T} d_t [f_{\xi(i,t,w_t)} - \sum_{j\in\mathcal{P}_t^i(w_t)\setminus\{i,\xi(i,t,w_t)\}} z_j]^+,$$

where w_t is the single element in $W(i)$ for each $t(p)+1 \leq t \leq T$. Thus, (5–21) holds.

For Condition 2, $t(i) \leq t(p)$, we have

$$s_i = \sum_{t=t(i)+1}^{T} d_t m_t^i = d_{t(i)+1} m_{t(i)+1}^i + \sum_{t=t(i)+2}^{T} d_t m_t^i$$
$$\geq d_{t(i)+1}[f_{i+1}]^+ + \sum_{t=t(i)+2}^{T} d_t [m_t^{i+1} - z_{i+1}]^+, \quad \text{if } t(i) < t(p) \tag{5–23}$$
$$\geq d_{t(i)+1}[f_{q_{w_{t(i)+1}}}]^+ + \sum_{t=t(i)+2}^{T} d_t [m_t^{q_{w_t}} - z_{q_{w_t}}]^+, \quad \text{if } t(i) = t(p), \tag{5–24}$$

where (5–23) is based on (5–11), (5–12), and the nonnegativity of m_t^i and (5–24) is based on (5–13), (5–14), and the nonnegativity of m_t^i.

If $t(i) = t(p)$, based on (5–10)

$$[m_t^{q_{w_t}} - z_{q_{w_t}}]^+ \geq [f_{\xi(q_{w_t},t,w_t)} - \sum_{j\in\mathcal{P}_t^{q_{w_t}}(w_t)\setminus\{q_{w_t},\xi(q_{w_t},t,w_t)\}} z_j - z_{q_{w_t}}]^+$$
$$= [f_{\xi(q_{w_t},t,w_t)} - \sum_{j\in\mathcal{P}_t^{q_{w_t}}(w_t)\setminus\{\xi(q_{w_t},t,w_t)\}} z_j]^+$$
$$= [f_{\xi(p,t,w_t)} - \sum_{j\in\mathcal{P}_t^{p}(w_t)\setminus\{p,\xi(p,t,w_t)\}} z_j]^+,$$

where the second equation holds due to $\xi(q_{w_t}, t, w_t) = \xi(p, t, w_t)$, $\mathcal{P}_t^{q_{w_t}}(w_t) \subseteq \mathcal{P}_t^p(w_t)$ and $\mathcal{P}_t^{q_{w_t}}(w_t) \setminus \xi(q_{w_t}, t, w_t) = \mathcal{P}_t^p(w_t) \setminus \{p, \xi(q_{w_t}, t, w_t)\}$, because w_t is the single element in $W(p)$ for each $t(p)+1 \leq t \leq T$. Thus, (5–21) holds.

Now we only need to show that, if $t(i) < t(p)$,

$$[m_t^{i+1} - z_{i+1}]^+ \geq [f_{\xi(i,t,w_t)} - \sum_{j \in \mathcal{P}_t^i(w_t) \setminus \{i, \xi(i,t,w_t)\}} z_j]^+. \tag{5–25}$$

It is easy to observe that if $t \leq t(p)$, (5–25) holds based on (5–11) and (5–12). Hence we discuss $t = t(p) + 1$ and $t > t(p) + 1$ respectively.

(a) If $t = t(p) + 1$, then

$$[m_t^{i+1} - z_{i+1}]^+ \geq [m_t^p - \sum_{j=i+1}^{p} z_j]^+ \geq [f_{q_{w_t}} - \sum_{j=i+1}^{p} z_j]^+ = [f_{q_{w_t}} - \sum_{j \in \mathcal{P}_t^i(w_t) \setminus \{i, q_{w_t}\}} z_j]^+.$$

where the first inequality follows (5–12) and the second inequality follows (5–13).

(b) If $t > t(p) + 1$, then

$$\begin{aligned}
[m_t^{i+1} - z_{i+1}]^+ &\geq [m_t^p - \sum_{j=i+1}^{p} z_j]^+ \geq [m_t^{q_{w_t}} - \sum_{j \in \mathcal{P}_{t(p)+1}^i(w_t) \setminus \{i\}} z_j]^+ \\
&\geq [f_{\xi(i,t,w_t)} - \sum_{j \in \mathcal{P}_{t(p)+1}^i(w_t) \setminus \{i\}} z_j - \sum_{j \in \mathcal{P}_t^{q_{w_t}}(w_t) \setminus \{q_{w_t}, \xi(i,t,w_t)\}} z_j]^+ \\
&\geq [f_{\xi(i,t,w_t)} - \sum_{j \in \mathcal{P}_t^i(w_t) \setminus \{i, \xi(i,t,w_t)\}} z_j]^+,
\end{aligned}$$

where the first inequality follows (5–12), the second one follows (5–14), and the third one follows (5–10). Thus (5–21) holds.

Therefore, Claim 1 holds.

Now we prove Claim 2 holds.

For any give exteme point $(f, g, z, s, \ell) \in Q_M$, we construct m and n such that $(f, g, z, m, n, s, \ell) \in Q_H$. That is, (f, g, z, m, n, s, ℓ) satisfies the conditions (5–4), (5–5), (5–10) to (5–17). Now for a given point $(\widehat{f}, \widehat{g}, \widehat{z}, \widehat{s}, \widehat{\ell}) \in Q_M$, let

$$\widehat{m}_t^i = \max_{w \in W(i)} \left[\widehat{f}_{\xi(i,t,w)} - \sum_{j \in \mathcal{P}_t^i(w) \setminus \{i, \xi(i,t,w)\}} \widehat{z}_j \right]^+ \tag{5–26}$$

$$\text{and} \quad \widehat{n}_t^i = \left[\widehat{g}_{\eta(i,t)} - \sum_{j \in \mathcal{P}(\eta(i,t),i) \setminus \{\eta(i,t)\}} \widehat{z} \right]^+ . \tag{5–27}$$

Since $(\widehat{f}, \widehat{g}, \widehat{z}, \widehat{s}, \widehat{\ell})$ is an extreme point in Q_M, we first observe that $(\widehat{f}, \widehat{g}, \widehat{z}, \widehat{s}, \widehat{\ell})$ satisfies equation (5–15) based on (5–26) and (5–19). Similarly, $(\widehat{f}, \widehat{g}, \widehat{z}, \widehat{s}, \widehat{\ell})$ satisfies equation (5–17) based on (5–27) and (5–20). It is also obvious that $(\widehat{f}, \widehat{g}, \widehat{z}, \widehat{m}, \widehat{n}, \widehat{s}, \widehat{\ell})$ satisfies (5–4) and (5–5). Based on (5–26), inequalities (5–10), (5–11), and (5–13) hold. Based on (5–27) and the nonnegativity of $\widehat{n}_t^i$, inequality (5–16) holds.

Now we only need to show that (5–12) and (5–14) hold.

For (5–12), let w^* be the scenario where $\widehat{m}_t^{i+1}$ achieves the maximum value. Then we observe that $\widehat{m}_t^i$ achieves the maximum value in the same scenario w^*. We prove (5–12) holds based on $\widehat{z}_{i+1} = 1$ and $\widehat{z}_{i+1} = 0$ respectively. If $\widehat{z}_{i+1} = 1$, then $\widehat{m}_t^i = 0$ based on (5–26). Then $\widehat{m}_t^i = 0 \geq \widehat{m}_t^{i+1} - 1 = \widehat{m}_t^{i+1} - \widehat{z}_{i+1}$. If $\widehat{z}_{i+1} = 0$, then

$$\begin{aligned}
\widehat{m}_t^i &= \left[\widehat{f}_{\xi(i,t,w^*)} - \sum_{j \in \mathcal{P}_t^i(w^*) \setminus \{i, \eta(i,t,w^*)\}} \widehat{z}_j \right]^+ \\
&= \left[\widehat{f}_{\xi(i,t,w^*)} - \sum_{j \in \mathcal{P}_t^i(w^*) \setminus \{i, i+1, \eta(i,t,w^*)\}} \widehat{z}_j \right]^+ \\
&= \left[\widehat{f}_{\xi(i,t,w^*)} - \sum_{j \in \mathcal{P}_t^{i+1}(w^*) \setminus \{i+1, \eta(i,t,w^*)\}} \widehat{z}_j \right]^+ \\
&= \widehat{m}_t^{i+1} = \widehat{m}_t^{i+1} - \widehat{z}_{i+1}.
\end{aligned}$$

Thus, (5–12) holds.

For (5–14), according to (5–26),

$$\begin{aligned}
\widehat{m}_t^p &= \max_{w_t \in W(p)} \left[\widehat{f}_{\xi(p,t,w_t)} - \sum_{j \in \mathcal{P}_t^p(w_t) \setminus \{p, \xi(p,t,w_t)\}} \widehat{z}_j \right]^+ \\
&\geq \left[\widehat{f}_{\xi(p,t,w_t)} - \sum_{j \in \mathcal{P}_t^p(w_t) \setminus \{p, \xi(i,t,w_t)\}} \widehat{z}_j \right]^+ , \text{ for each } w_t \in W(p).
\end{aligned}$$

Now for each particular $w_t \in W(p)$,

$$
\begin{aligned}
& \left[\widehat{f}_{\xi(p,t,w_t)} - \sum_{j \in \mathcal{P}_t^p(w_t) \setminus \{p, \xi(p,t,w_t)\}} \widehat{z}_j \right]^+ \\
= & \left[\widehat{f}_{\xi(p,t,w_t)} - \sum_{j \in \mathcal{P}_t^p(w_t) \setminus \{p, q_{w_t}, \xi(p,t,w_t)\}} \widehat{z}_j - \widehat{z}_{q_{w_t}} \right]^+ \\
= & \left[\widehat{f}_{\xi(q_{w_t},t,w_t)} - \sum_{j \in \mathcal{P}_t^{q_{w_t}}(w_t) \setminus \{q_{w_t}, \xi(p,t,w_t)\}} \widehat{z}_j - \widehat{z}_{q_{w_t}} \right]^+ \\
= & \left[\widehat{m}_t^{q_{w_t}} - \widehat{z}_{q_{w_t}} \right]^+ \\
\geq & \ \widehat{m}_t^{q_{w_t}} - \widehat{z}_{q_{w_t}},
\end{aligned}
$$

where the second equation follows $\mathcal{P}_t^p(w_t) \setminus \{p, q_{w_t}\} = \mathcal{P}_t^{q_{w_t}}(w_t) \setminus \{q_{w_t}\}$ and the third equation follows $\widehat{m}_t^{q_{w_t}} = f_{\xi(q_{w_t},t,w_t)} - \sum_{j \in \mathcal{P}_t^{q_{w_t}}(w_t) \setminus \{q_{w_t}, \xi(p,t,w_t)\}} \widehat{z}_j$ for a given $w_t \in W(p)$. Then (5–14) holds. Thus $(\widehat{f}, \widehat{g}, \widehat{z}, \widehat{m}, \widehat{n}, \widehat{s}, \widehat{\ell}) \in Q_H$ and Claim 2 holds.

Therefore, the conclusion holds. □

Table 5-1. The matrix of constraints (5–4), (5–5), and (5–10) to (5–17) for the example in Figure 5-1

	z_1	z_2	z_3	z_4	z_5	f_1	f_2	f_3	f_4	f_5	g_1	g_2	g_3	g_4	g_5	m_2^1	m_3^1	m_3^2	m_3^3	n_1^2	n_1^3	n_2^4	n_1^4	n_2^5	n_1^5
1	1					1					1														
2		1					1					1													
3			1					1					1												
4				1					1					1											
5					1					1					1										
6		1									-1	1													
7			1								-1		1												
8				1								-1		1											
9					1								-1		1										
10									-1									1							
11										-1									1						
12							-1									1									
13								-1								1									
14		1															1	-1							
15			1														1		-1						
16		1									-1									1					
17			1								-1										1				
18				1								-1										1			
19					1								-1											1	
20		1		1							-1												1		
21			1		1						-1														1

CHAPTER 6
LIFTING SCHEME FOR THE STOCHASTIC DYNAMIC KNAPSACK POLYTOPE

6.1 Introduction

Deterministic dynamic knapsack set is naturally generated from the deterministic uncapacitated lot-sizing problem. Loparic et al. (2003) first introduced the deterministic dynamic knapsack set X_{DK} as follows:

$$X_{DK} = \left\{ (s, y) \in \mathbb{R}_+ \times \mathbb{B}^T : s + \sum_{\tau=1}^{t} a_\tau y_\tau \geq b_t, \, t \in T \right\}, \tag{6–1}$$

where $a, \; b \in \mathbb{R}_+^T$. They studied the polyhedral structure of the deterministic dynamic knapsack set. With the application of the sequence independent lifting scheme, a family of facet-defining inequalities for X_{DK} were introduced.

For the deterministic mixed-integer programming problems, different schemes have been explored to generate more valid inequalities based on existing valid inequalities. Guan et al. (2007) proposed the pairing scheme which generated a family of inequalities with the properly ordered combination of two existing valid inequalities. Recently, Günlük and Pochet (2001) developed the mixing procedure to generate valid inequalities based on the mixed-integer rounding inequalities (MIR). They demonstrated that the mixing inequalities are strong inequalities for some special polyhedral structures. Miller and Wolsey (2003) showed that the mixing inequalities can provide the convex hull description for a special single-item lot-sizing problem.

The lifting scheme was first introduced by Wolsy (1976, 1977), and Zemel (1978), et al. The lifting scheme can be applied sequentially to generate strong valid inequalities. Gu et al. (2000) extended the sequence independent lifting scheme (Wolsy 1977) to the mixed 0-1 integer programming, and showed that the sequence independent lifting property holds as long as the lifting function is superadditive. That is the generation of lifting coefficients are independent of the lifting sequence. Atamtürk (2004) generalized

the sequence independent lifting property to the general mixed-integer programming problem.

In this chapter, we study the extension of the deterministic dynamic knapsack set: the stochastic dynamic knapsack (SDK) set. We investigate the polytope of the SDK set based on a multi-stage stochastic scenario tree model as described in Ruszczyński and Shapiro (2003). For the stochastic mixed-integer programming problems, Guan et al. (2006b) studied the multi-stage stochastic uncapacitated lot-sizing problem and developed a family of valid inequalities, named $(\mathcal{Q}, S_{\mathcal{Q}})$ inequalities. They showed that under certain conditions, $(\mathcal{Q}, S_{\mathcal{Q}})$ inequalities are facet-defining inequalities. Guan et al. (2006a) examined that $(\mathcal{Q}, S_{\mathcal{Q}})$ inequalities are sufficient to describe the convex hull of the two-period problem. The pairing scheme described in Guan et al. (2007) can also provide valid inequalities for the stochastic lot-sizing problem and generalize all $(\mathcal{Q}, S_{\mathcal{Q}})$ inequalities. Further more, based on these, Guan et al. (2009) proposed a general approach to generate valid inequalities for the multi-stage stochastic mixed-integer programming problems. They combined valid deterministic inequalities corresponding to each scenarios to generate valid inequalities for the whole scenario tree.

The remaining part of this chapter is organized as follows. In Section 6.3, we study the pairing and mixing schemes for the SDK set and show that pairing and mixing inequalities are facet-defining inequalities under certain conditions. Section 6.4 demonstrates the sequence independent and sequence dominant lifting schemes for the SDK set. We generate families of valid inequalities for the SDK set through the lifting schemes. Then, to solve large-scale problem, in Section 6.5, we apply parallel computing technique to solve the SDK set. We develop parallel algorithms to generate valid inequalities via pairing, mixing, and lifting schemes for the stochastic capacitated lot-sizing problem as an example of the SDK set. Finally, we demonstrate the computational efficiency by showing the improvement of the optimality and integrality gaps in Section6.6.

6.2 The Path Inequality

With the notation of stochastic scenario tree in Chapter 1, we extend X_{DK} to a stochastic dynamic knapsack set with a single continuous variable defined by:

$$X_{SDK} = \left\{ (s, y) \in \mathbb{R}_+ \times \mathbb{B}^n : s + \sum_{j \in \mathcal{P}(i)} a_j y_j \geq b_i, i \in \mathcal{V} \right\},$$

where $n = |\mathcal{V}|$, $a \in \mathbb{R}^n_+$, $b \in \mathbb{R}^n$. In the following sections, without loss of generality, we assume $b_j \leq b_i$, if $j \in \mathcal{P}(i)$.

Similar as the deterministic case, X_{SDK} is a relaxation set of the feasible region of stochastic capacitated lot-sizing problem, X_{CSLS}.

$$X_{CSLS} = \left\{ (s, x, y) \in \mathbb{R}_+ \times \mathbb{R}^n_+ \times \mathbb{B}^n : s + \sum_{j \in \mathcal{P}(i)} x_j \geq d_{1i}, x_i \leq a_i y_i, i \in \mathcal{V} \right\},$$

where d_i and a_i are the demand and capacity at time period i, respectively, and $d_{1i} = \sum_{j \in P(i)} d_j$. Replacing x_i with $a_i y_i$, we get the relaxation set of X_{CSLS} as

$$X_{RSLS} = \left\{ (s, y) \in \mathbb{R}_+ \times \mathbb{B}^n : s + \sum_{j \in \mathcal{P}(i)} a_j y_j \geq b_i, i \in \mathcal{V} \right\},$$

where $b_i = \max_{j \in \mathcal{P}(i)} \{d_{1j}\}$. It is obvious that X_{RSLS} is a X_{SDK} set.

X_{SDK} evolves the information of the whole scenario tree. Now let $X(i)$ represent the path set of X_{SDK} for a given node $i \in \mathcal{V}$, i.e.,

$$X(i) = \{s + \sum_{k \in \mathcal{P}(j)} a_k y_k \geq b_j, \quad j \in \mathcal{P}(i)\}, \tag{6–2}$$

We can observe that $X(i)$ is a $t(i)$-period deterministic dynamic knapsack set corresponding to the scenario with the information on path $\mathcal{P}(i)$. We name the inequality in $X(i)$ as path inequality. All path inequalities in $X(i)$, $i \in \mathcal{V}$ are valid for X_{SDK}. Note here, when $|\mathcal{L}| = 1$, $X_{SDK} = X_{DK}$.

Proposition 6.1. *X_{SDK} can be represented by the joined set of $X(i)$ for all $i \in \mathcal{V}$, i.e.,*

$$X_{SDK} = \bigcap_{i \in \mathcal{V}} X(i).$$

Now we show an example of path inequalities, based on Figure 6-1. The following path inequalities are generated corresponding to path $\mathcal{P}(1)$, $\mathcal{P}(2)$, $\mathcal{P}(3)$, $\mathcal{P}(4)$, and $\mathcal{P}(5)$ separately:

$$\begin{array}{lllllll}
s & +40y_1 & & & & & \geq 5 \\
s & +40y_1 & +15y_2 & & & & \geq 15 \\
s & +40y_1 & & +20y_3 & & & \geq 17 \\
s & +40y_1 & & +20y_3 & +20y_4 & & \geq 20 \\
s & +40y_1 & & +20y_3 & & 20y_5 & \geq 20
\end{array}$$

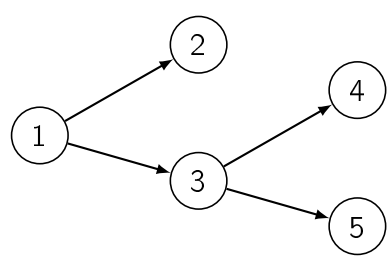


Figure 6-1. Path inequalities for a scenario tree

Because $X(i)$ is a $t(i)$-period deterministic dynamic knapsack set, we extend the conclusion of Loparic et al. (2003) for X_{DK} to the stochastic setting. We let $b_{\ell k} = \sum_{t \in \mathcal{P}(\ell,k) \subseteq \mathcal{P}(j)} b_t$ and $\widetilde{b}_k = b_{1k}$. With the information of b, we construct a modified nonnegative parameter $\overline{b}$. Let $\overline{b}_j = \max_{k \in \mathcal{P}(j)} \{\widetilde{b}_k, 0\}$. We order nodes in path $\mathcal{P}(\ell, j)$ as $i_1, \cdots, i_k, i_{k+1}, \cdots, i_K$ with $K = t(j) - t(\ell) + 1$.

Proposition 6.2. *The inequality*

$$s + \sum_{\ell \in \mathcal{P}(j)} \overline{\phi}_\ell(\mathcal{R}) y_\ell \geq \overline{b}_j$$

is valid for $conv(X(i))$ for $j \in \mathcal{P}(i)$, and facet-defining when $j = i$, where $\mathcal{R} = \mathcal{P}(j)$, $\overline{\phi}_\ell(\mathcal{R}) = \min\{a_\ell, \sum_{i_k \in \mathcal{R}(\ell)} (\overline{b}_{i_k} - \overline{b}_{i_{k-1}})\}$ with $\mathcal{R}(\ell) = \mathcal{R} \cap \mathcal{V}(\ell)$.

In the following sections, we apply the pairing and lifting schemes to generate valid and strong inequalities for X_{SDK}.

6.3 The Pairing and Mixing Schemes for the Stochastic Dynamic Knapsack Set

In this section, we apply the pairing and mixing schemes to the path inequalities for X_{SDK} and generate corresponding tree inequalities. We show that these generated tree inequalities are facet-defining inequalities for X_{SDK} under certain conditions.
We introduce the notation of the vector operation for the following discussion. Given two same dimension vectors a^1 and a^2, $\min(a^1, a^2)$ and $\max(a^1, a^2)$ are carried out component-wise. For brevity, given a vector a and a scalar c, we let $\min\{a, c\} = \min\{a, c\mathbb{1}\}$, where $\mathbb{1}$ is a vector of ones of the same dimension as a.

Guan et al. (2007) studied the pairing scheme for the mixed-integer sets $X \in \mathbb{Z}_+^n \times \mathbb{R}_+^p$. Let the pair $(a, g) \in \mathbb{R}^{n+1} \times \mathbb{R}^p$ define a valid inequality for X, if

$$\sum_{i=1}^{n} a_i y_i + \sum_{j=1}^{p} g_j x_j \geq a_{n+1}, \text{ for all } (y, x) \in X. \tag{6–3}$$

Let (a^1, g^1) and (a^2, g^2) define two valid inequalities for X. The pairing scheme for these two inequalities is as follows:

Theorem 6.1. *(Guan et al. 2007) If (a^1, g^1) and (a^2, g^2) define two valid inequalities for X with $a^1_{n+1} \leq a^2_{n+1}$, the inequality*

$$\sum_{k=1}^{n} \phi_k y_k + \sum_{\ell=1}^{p} \psi_\ell x_\ell \geq a^2_{n+1} \tag{6–4}$$

is valid for X, where $\phi = \min\{a^1 + (a^2_{n+1} - a^1_{n+1})\mathbb{1}, \max\{a^1, a^2\}\}$, $\psi = \max(g^1, g^2)$.

Guan et al. (2009) applied the above pairing scheme to path inequalities of X_{SDK} and obtained the following tree inequalities:

Theorem 6.2 (Guan et al. 2009)**.** *Given a set $\mathcal{R} = \{i_1, i_2, \cdots, i_K\} \subseteq \mathcal{V}$ indexed such that $b_{i_1} \leq b_{i_2} \leq \cdots \leq b_{i_K}$, the inequality*

$$s + \sum_{j \in \mathcal{V}_{\mathcal{R}}} \phi_j(\mathcal{R}) y_j \geq b_{i_K} \tag{6–5}$$

where $\mathcal{V}_{\mathcal{R}} = \cup_{i_k \in \mathcal{R}} \mathcal{P}(i_k)$ *and* $\phi_j(\mathcal{R}) = \min\{a_j, \sum_{i_k \in \mathcal{R}(j)} (b_{i_k} - b_{i_{k-1}})\}$ *with* $\mathcal{R}(j) = \mathcal{R} \cap \mathcal{V}(j)$ *and* $b_0 = 0$*, is valid for* X_{SDK}*.*

Tree inequalities (6–5) can be strengthened as follows:

Theorem 6.3 (Guan et al. 2009)**.** *Given a set* $\mathcal{R} = \{i_1, i_2, \cdots, i_K\} \subseteq \mathcal{V}$ *indexed such that* $b_{i_1} \leq b_{i_2} \leq \cdots \leq b_{i_K}$*, let* $i'_k = argmin\{u : u \in \mathcal{P}(i_k)$ *and* $b_u \geq b_{i_{k-1}}\}$ *for each* $i_k \in \mathcal{R}$*,* $\Omega = \cup_{i_k \in \mathcal{R}} \mathcal{P}(i'_k, i_k)$*, and* $\Omega(j) = \Omega \cap \mathcal{V}(j)$*. Then, the inequality (6–5) is dominated by the inequality*

$$s + \sum_{j \in \mathcal{V}_\Omega} \phi_j(\Omega) y_j \geq b_{i_K} \tag{6–6}$$

where $\mathcal{V}_\Omega = \cap_{i_k \in \Omega} \mathcal{P}(i_k)$ *and* $\phi_j(\Omega) = \min\{a_j, \sum_{i_k \in \Omega(j)} (b_{i_k} - b_{i_{k-1}})\}$ *with* $b_0 = 0$*.*

With a stronger assumption of the coefficient, (6–6) is a facet-defining inequality for $conv(X_{SDK})$.

Theorem 6.4 (Guan et al. 2009)**.** *Inequality (6–6) is facet-defining for* $conv(X_{SDK})$ *if*

(1) $b_{i_K} = \max\{b_i : i \in \mathcal{V}\}$*.*

(2) *for each* $j \in \mathcal{V}_\Omega$*,* $a_j \geq \max\{b_i, i \in \Omega(j)\}$*.*

Guan et al. (2009) discussed the convex hull description of X_{SDK} under the large coefficient case (Condition (2) in Theorem 6.4).

Theorem 6.5 (Guan et al. 2009)**.** *If* $a_j \geq \max\{b_k, k \in \mathcal{V}(j)\}$ *for each* $j \in \mathcal{V}$*, then the family of inequalities (6–6) for all* $\Omega \subset \mathcal{V}$*, together with* $0 \leq s \leq b_{\mathcal{V}}$ *and* $0 \leq y_j \leq 1$ *for each* $j \in \mathcal{V}$*, describe the convex hull of* X_{SDK}*, where* $b_{\mathcal{V}} = \max\{b_i, i \in \mathcal{V}\}$*.*

They also discussed the separation algorithm of tree inequalities for X_{SDK}. Based on the shortest path algorithm, the corresponding separation algorithm is polynomial.

Theorem 6.6 (Guan et al. 2009)**.** *If* $a_j \geq \max\{b_k, k \in \mathcal{V}(i)\}$ *for all* $i \in \mathcal{V}$*, there exists a polynomial-time separation algorithm for the tree inequalities (6–6).*

We show an example of generating the tree inequality (6–4) based on the scenario tree in Figure 6-1. For set $\mathcal{R} = \{1, 2\}$, we have $\mathcal{V}_R = \{1, 2\}$, $\mathcal{R}(1) = \{1, 2\}$, and $\mathcal{R}(2) = \{2\}$. The corresponding two path inequalities are $s + 40y_1 \geq 5$ and $s + 40y_1 + 15y_2 \geq 15$.

Then,

$$\begin{aligned}
\phi_1(\mathcal{R}) &= \min\{\max(a_{11}, a_{12}, b_1 - b_0 + b_2 - b_1)\} \\
&= \min\{\max(40, 40), 5 - 0 + 15 - 5\} = 15, \quad \text{and} \\
\phi_2(\mathcal{R}) &= \min\{\max(a_{12}, a_{22}, b_2 - b_1\} \\
&= \min\{\max(40, 40), 15 - 5\} = 10.
\end{aligned}$$

The corresponding pairing inequality is

$$s + 15y_1 + 10y_2 \geq 15.$$

Next, we apply the mixing scheme to X_{SDK} and generate tree inequalities.

Günlük and Pochet (2001) proposed the mixing scheme for MIR inequalities. Given a mixed-integer region $S \subseteq R^{m_1} \times Z^{m_2}$ and a collection of $m \geq 2$ valid inequalities for S

$$f^i(x) + Bg^i(x) \geq \pi^i, \ i \in \mathcal{I} = \{1, \ldots, m\}, \tag{6–7}$$

where $B \in R^1_+$, $\pi^i \in R^1$, $f^i(x) \geq 0$, and $g^i(x) \in Z$. For any $i \in \mathcal{I}$, the simple MIR inequality

$$f^i(x) \geq \gamma^i(\tau^i - g^i(x))$$

is valid for S, where $\tau_i = \lceil \pi^i / B \rceil$, $\gamma^i = \pi^i - (\tau^i - 1)B$, $\tau^i \in Z^1$, and $B \geq \gamma^i > 0$. Note that f^i and g^i can be nonlinear, π^i and g^i can be negative. Without loss of generality, we assume that $I = 1, \ldots, n$, and $\gamma^i \geq \gamma^{i-1}$ for all $n \geq i \geq 2$.

Theorem 6.7 (Günlük and Pochet 2001)**.** *The following two inequalities*

$$\overline{f}(x) \geq \sum_{i=1}^{n} (\gamma^i - \gamma^{i-1})(\tau^i - g^i(x)) \tag{6–8}$$

and

$$\overline{f}(x) \geq \sum_{i=1}^{n} (\gamma^i - \gamma^{i-1})(\tau^i - g^i(x)) + (B - \gamma^n)(\tau^1 - g^1(x) - 1) \tag{6–9}$$

with $\gamma^0 = 0$, are valid for S, where $\overline{f}(x) \geq f^i(x) \geq 0$ for all $x \in S$ and $i \in I$.

Now we apply the mixing scheme to X_{SDK}, and prove that the mixing inequalities can provide the convex hull description of X_{SDK} under certain conditions. Let $z_i = \sum_{j \in P(i)} y_j$ and $A = \max\{a_j, j \in P(i)\}$. Then,

$$X'_{SDK} = \{(w, z) \in \mathbb{R} \times \mathbb{Z}^{|\mathcal{V}|}, s + Az_i \geq b_i, i \in \mathcal{V}\} \tag{6–10}$$

is a relaxation set of X_{SDK}. We generate the mixing inequalities for X'_{SDK} as follows:

Proposition 6.3. *The following two inequalities*

$$s \geq \sum_{i=1}^{n} (\gamma^i - \gamma^{i-1})(\tau^i - z_i) \tag{6–11}$$

and

$$s \geq \sum_{i=1}^{n} (\gamma^i - \gamma^{i-1})(\tau^i - z_i) + (B - \gamma^n)(\tau^1 - z_1 - 1) \tag{6–12}$$

are valid for X'_{SDK}, where $\gamma^0 = 0$, $\tau^i = \lceil b_i/A \rceil$ and $\gamma^i = b_i - (\tau^i - 1)A$.

Replacing z_i by $\sum_{j \in P(i)} y_j$, we obtain the valid inequalities for X_{SDK} by the mixing scheme.

Proposition 6.4. *The following two inequalities*

$$s \geq \sum_{i=1}^{n} (\gamma^i - \gamma^{i-1})(\tau^i - \sum_{j \in P(i)} y_j) \tag{6–13}$$

and

$$s \geq \sum_{i=1}^{n} (\gamma^i - \gamma^{i-1})(\tau^i - \sum_{j \in P(i)} y_j) + (B - \gamma^n)(\tau^1 - y_1 - 1) \tag{6–14}$$

are valid for X'_{SDK}, where $\gamma^0 = 0$, $\tau^i = \lceil b_i/A \rceil$ and $\gamma^i = b_i - (\tau^i - 1)A$.

Note here, the mixing inequalities (6–11) and (6–12) can also be constructed for subset $X_{SDK}(\mathcal{V}(i))$ based on the subtree $\mathcal{V}(i)$.

We provide the convex hull description for X_{SDK} with mixing inequalities (6–13) and (6–14) under the condition that $a_i = A$ for $i \in \mathcal{V}$.

Theorem 6.8. *If $a_i = A$ for $i \in \mathcal{V}$, the family of inequalities (6–13) and (6–14) are sufficient to describe the convex hull for X_{SDK}.*

Proof. We prove this conclusion by two claims.

Claim 1. Inequalities (6–11), (6–12), and $0 \leq z_i - z_{a(i)} \leq 1$ define the convex hull description for

$$X' = \{(w, z) \in \mathbb{R} \times \mathbb{Z}_+^{|\mathcal{V}|}, w + Az_i \geq f_i, 0 \leq z_i - z_{a(i)} \leq 1, i \in \mathcal{V}\}. \tag{6–15}$$

Claim 2. Let X'_S denote X_{SDK} with $a_i = A$ for $i \in \mathcal{V}$. It is one to one correspondence between X'_S and X'.

Proof of Claim 1: Let $Z_1 = \{(w, z) \in \mathbb{R} \times \mathbb{Z}_+^{|\mathcal{V}|}, w + Az_i \geq f_i, i \in \mathcal{V}\}$ and $Z_2 = \{z \in \mathbb{Z}_+^{|\mathcal{V}|}, 0 \leq z_i - z_{a(i)} \leq 1, i \in \mathcal{V}\}$. We have $X' = Z_1 \cap Z_2$. The constraint matrix for Z_2 is the transpose of a network matrix (arc-node incidence matrix) and the right-hand coefficient is integer. Following the result in Miller and Wolsey (2003), (6–11), (6–12), and $0 \leq z_i - z_{a(i)} \leq 1$ are sufficient to provide the convex hull description for X'.

Proof of Claim 2: we show that for a given $(s, y) \in X'_S$, there exists a reversible function $G : X'_S \to X'$.

We show that there is a function $G : X'_S \to X'$, where

$$w = s, \tag{6–16}$$

$$z_i = \sum_{j \in \mathcal{P}(i)} y_j, \quad i \in \mathcal{V}. \tag{6–17}$$

Because $y_i \in \{0, 1\}$, we have $z_i \in \mathbb{Z}_+$. Let $f_i = b_i$ for all $i \in \mathcal{V}$. With $s + A\sum_{j \in \mathcal{P}(i)} y_j \geq b_i$, (6–16), and (6–17), we have $w + Az_i \geq f_i$. Thus, $(w, z) \in X'$.

The reverse function of G, $G^{-1} : X' \to X'_S$, is that

$$s = w, \tag{6–18}$$

$$y_i = z_i - z_{a(i)}, \quad i \in \mathcal{V} \tag{6–19}$$

Because $z_i \in \mathbb{Z}_+$ and $0 \leq z_i - z_{a(i)} \leq 1$, we have $y_i \in \{0, 1\}$. With $w + Az_i \geq f_i$, (6–18), and (6–19), we have $s + A\sum_{j \in \mathcal{P}(i)} y_j \geq f_i$. Thus, $(s, y) \in X'_S$.

Thus, it is one to one correspondence between X'_S and X'.

Finally, we show that (6–13) and (6–14) are sufficient to provide the convex hull description for X'_S. Given any $c_1 \in \mathbb{R}$ and $c_2 \in \mathbb{R}^{|V|}$, the problem, P1, $\max\{c_1 s + c_2^T y, (s, y) \in \mathbb{R} \times \mathbb{R}_+^{|\mathcal{V}|}\}$ with (6–13) and (6–14) can be transferred to be the problem, P2, $\max\{c_1 w + b^T z, (w, z) \in \mathbb{R} \times \mathbb{R}_+^{|\mathcal{V}|}\}$ with (6–11), (6–12), and $0 \leq z_i - z_{a(i)} \leq 1$, where $b_i = \sum_{j \in \mathcal{P}(i)} c_2^j$. Suppose that (w^*, z^*) is the optimal solution for P2. With Claim 1, z^* is integral. With Claim 2 and (w^*, z^*), we obtain the corresponding optimal solution (s^*, y^*) for P1. With the integrality of z^* and $0 \leq y_i = z_i - z_{a(i)} \leq 1$, y^* is integral. Thus, (6–13) and (6–14) are sufficient to provide the convex hull description for X'_S. □

6.4 The Lifting Scheme for the Stochastic Dynamic Knapsack Set

In this section, we derive more valid inequalities for X_{SDK} by the lifting scheme. First, we apply the lifting scheme to the path inequalities and show that the sequence independent lifting property holds when we lift back variables based on path inequalities. Second, we discuss the sequence dominant lifting procedure for the tree inequality in X_{SDK}. Third, we discuss how to combine pairing and lifting schemes and apply them to the two-stage stochastic dynamic knapsack set and the multi-stage stochastic dynamic knapsack set.

Similar as Loparic et al. (2003), we derive other valid inequalities for X_{SDK} by the lifting scheme. We set some variables to be 1 and modify the corresponding parameters. Then, we generate the basic inequality and lift back variables that have been fixed.

Let $\mathcal{U}(i)$ represent a subset of $\mathcal{P}(i)$. We set $y_\ell = 1$ for $\ell \in \mathcal{U}(i)$. Then we modify the parameter b_ℓ as $\widetilde{d}_\ell = b_\ell - a_\ell$, for $\ell \in \mathcal{U}(i)$, and generate the basic path inequality as follows:

$$s + \sum_{j \in \mathcal{P}(i) \setminus \mathcal{U}(i)} \overline{\phi}_j(\mathcal{R}) y_j \geq \overline{b}_i \tag{6–20}$$

where $\mathcal{R} = \mathcal{P}(i)$, $\overline{\phi}_j(\mathcal{R}) = \min\{a_j, \sum_{i_k \in \mathcal{R}(j)} (\overline{b}_{i_k} - \overline{b}_{i_{k-1}})\}$, and $\overline{b}_i = \max\{\max_{t \in \mathcal{P}(i)} (\sum_{j \in \mathcal{P}(t)} \widetilde{d}_j), 0\}$.

We define the function

$$\xi_{\mathcal{U}(i)}(h) = \min\{s + \sum_{j \in \mathcal{P}(i) \setminus \mathcal{U}(i)} \overline{\phi}_j(\mathcal{R}) y_j - \overline{b}_i : (s, y) \in X_{\mathcal{P}(i) \setminus \mathcal{U}(i)}(\widetilde{d} + h)\} \tag{6–21}$$

With (6–20) and (6–21) as the basic inequality and the lifting function, respectively, we generate the lifted valid inequality for X_{SDK} as follows:

Proposition 6.5. *The inequality*

$$s + \sum_{j \in P(i) \setminus \mathcal{U}(i)} \overline{\phi}_j(\mathcal{R}) y_j + \sum_{j \in \mathcal{U}(i)} \xi_{\mathcal{U}(i)}(a_j e^j) y_j \geq \overline{b}_i + \sum_{j \in \mathcal{U}(i)} \xi_{\mathcal{U}(i)}(a_j e^j)$$

is valid for X_{SDK} with $\mathcal{R} = \mathcal{P}(i)$ and facet-defining for $X(i)$, where $i \in \mathcal{V}$.

Proof. We let $K = |\mathcal{P}(i)|$ and re-index nodes on $\mathcal{P}(i)$ from root node to node i as $\{1, 2, \cdots, K\}$. Then $X(i) = \{(s, y) \in \mathbb{R} \times \{0, 1\}^K, s + \sum_{j=1}^{t} a_j y_j \geq b_t, t = 1, \cdots, K\}$. $X(i)$ is a deterministic dynamic knapsack set. We apply the lifting scheme to $s + \sum_{j=1}^{i} a_j y_j \geq b_i$ and re-index node index in (6–20) and (6–21). Directly following Loparic et al. (2003), we obtain the above result. □

Note that based on Proposition 6.5, the sequence independent lifting property holds for lifting back variables based on the path inequality in X_{SDK}.

Now we derive valid inequalities for X_{SDK} based on the tree inequality. Let $X_{SDK}(\mathcal{V}_{\mathcal{R}})$ represent the feasible solution set defined by the path inequalities corresponding to nodes in $\mathcal{V}_{\mathcal{R}}$. We let $\mathcal{L}(\mathcal{R})$ represent the set of leaf node in $\mathcal{V}_{\mathcal{R}}$ and $\mathcal{C}_{\mathcal{R}}(i) = \cup_{j \in \mathcal{P}(i)} \mathcal{C}(j) \setminus \mathcal{V}_{\mathcal{R}}$ with all $i \in \mathcal{L}(\mathcal{R})$.

First, we apply the lifting scheme to $X_{SDK}(\mathcal{V}_{\mathcal{R}})$. Suppose that

$$s + \sum_{j \in \mathcal{V}_{\mathcal{R}}} \beta_j y_j \geq \gamma \tag{6–22}$$

is a facet-defining inequality. We set $y_\ell = 0$ for a given $\ell \in \mathcal{C}_\mathcal{R}(i)$ and let (6–22) be the basic inequality. We define the lifting function as

$$\xi_{C_{\mathcal{R}(i)}}(0) = \min\{s + \sum_{j \in \mathcal{V}_\mathcal{R}} \beta_j y_j - \gamma : (s, y) \in X_{SDK}(\mathcal{V}_\mathcal{R}) \cup \{\ell\}\}$$

Then, we obtain the following result:

Lemma 1. *If $s + \sum_{j \in \mathcal{V}_\mathcal{R}} \beta_j y_j \geq \gamma$ is a facet-defining inequality for $X_{SDK}(\mathcal{V}_\mathcal{R})$, then corresponding to each node $i \in \mathcal{L}(\mathcal{R})$,*

$$s + \sum_{j \in \mathcal{V}_\mathcal{R}} \beta_j y_j + \beta_\ell y_\ell \geq \gamma + \beta_\ell \tag{6–23}$$

is facet-defining for $X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell\})$ for each $\ell \in \mathcal{C}_\mathcal{R}(i)$, where $\beta_\ell = \min\{s + \sum_{j \in \mathcal{V}_\mathcal{R}} \beta_j y_j - \gamma : (s, y) \in X_{SDK}(\mathcal{V}_\mathcal{R}) \cup \{\ell\}\}$.

Second, we consider two variables y_{ℓ_k} and $y_{\ell_{k-1}}$. If they satisfy $\ell_k, \ell_{k-1} \in \mathcal{C}_\mathcal{R}(i)$ for a given $i \in \mathcal{L}(\mathcal{R})$, $b_{\ell_k} \geq b_{\ell_{k-1}}$, and $a(\ell_k) \in \mathcal{P}(a(\ell_{k-1}))$, we discuss the following two conditions to show the relationship between the coefficients of y_{ℓ_k} and $y_{\ell_{k-1}}$ in the lifted valid inequality, and the sequence of lifting y_{ℓ_k} and $y_{\ell_{k-1}}$.

Condition 1: y_{ℓ_k} is lifted before $y_{\ell_{k-1}}$. With Lemma 1, after lifting back of y_{ℓ_k}, we obtain a facet-defining inequality (6–23) for $X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\})$ and the coefficient of y_{ℓ_k} is $\beta_{\ell_k} = \min\{s + \sum_{j \in \mathcal{V}_\mathcal{R}} \beta_j y_j - \gamma : (s, y) \in X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\})\}$. Then, we lift $y_{\ell_{k-1}}$ based on (6–23) and obtain a facet-defining inequality for $X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\} \cup \{\ell_{k-1}\})$, $s + \sum_{j \in \mathcal{V}_\mathcal{R}} \beta_j y_j + \beta_{\ell_k} y_{\ell_k} + \beta_{\ell_{k-1}} y_{\ell_{k-1}} \geq \gamma + \beta_{\ell_k} + \alpha_{\ell_{k-1}}$ with

$$\alpha_{\ell_{k-1}} = \min\{s + \sum_{j \in \mathcal{V}_\mathcal{R}} \beta_j y_j + \beta_{\ell_k} y_{\ell_k} - (\gamma + \beta_{\ell_k}) : (s, y) \in X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\} \cup \{\ell_{k-1}\})\}.$$

In order to obtain the value of $\beta_{\ell_{k-1}}$, we show that for any $(s, y) \in X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\})$ is feasible for $X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\} \cup \{\ell_{k-1}\})$, where $X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\} \cup \{\ell_{k-1}\}) = X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\}) \cup \{(s, y), s + \sum_{j \in \mathcal{P}(\ell_{k-1})} a_j y_j \geq b_{\ell_{k-1}}\}$. With an unbounded variable s,

$y_{\ell_k} = 0$ is feasible for $X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\})$. Then, any $(s, y) \in X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\})$ satisfies

$$s + \sum_{j \in \mathcal{P}(\ell_k) \setminus \{\ell_k\}} a_j y_j \geq b_{\ell_k}.$$

Due to $a(\ell_k) \in \mathcal{P}(a(\ell_{k-1}))$, we have $\mathcal{P}(\ell_k) \setminus \{\ell_k\} \subseteq \mathcal{P}(\ell_{k-1}) \setminus \{\ell_{k-1}\}$ and

$$\begin{aligned}
& s + \sum_{j \in \mathcal{P}(\ell_{k-1})} a_j y_j \\
=& s + \sum_{j \in \mathcal{P}(\ell_{k-1}) \setminus \{\ell_{k-1}\}} a_j y_j + a_{\ell_{k-1}} y_{\ell_{k-1}} \\
=& s + \sum_{j \in \mathcal{P}(\ell_k) \setminus \{\ell_k\}} a_j y_j + \sum_{j \in (\mathcal{P}(\ell_{k-1}) \setminus \{\ell_{k-1}\}) \setminus (\mathcal{P}(\ell_k) \setminus \{\ell_k\})} a_j y_j + a_{\ell_{k-1}} y_{\ell_{k-1}} \\
\geq& b_{\ell_k}.
\end{aligned}$$

With $b_{\ell_k} \geq b_{\ell_{k-1}}$, we have $s + \sum_{j \in \mathcal{P}(\ell_{k-1})} a_j y_j \geq b_{\ell_{k-1}}$ for any $(s, y) \in X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\})$. Therefore, all feasible solutions for $X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\})$ are feasible solutions for $X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\} \cup \{\ell_{k-1}\})$. Because variable s does not have an upper bound, (s, y) with $y_{\ell_k} = 0$ is a feasible solution for $X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\} \cup \{\ell_{k-1}\})$. Then, we have

$$\begin{aligned}
\alpha_{\ell_{k-1}} &= \min\{s + \sum_{j \in \mathcal{V}_\mathcal{R}} \beta_j y_j + \beta_{\ell_k} y_{\ell_k} - (\gamma + \beta_{\ell_k}) : (s, y) \in X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\}) \cup \{\ell_{k-1}\}\} \\
&= \min\{s + \sum_{j \in \mathcal{V}_\mathcal{R}} \beta_j y_j - \gamma + 0\beta_{\ell_k} - \beta_{\ell_k} : (s, y) \in X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_k\}) \cup \{\ell_{k-1}\}\} \\
&= \beta_{\ell_k} - \beta_{\ell_k} = 0.
\end{aligned}$$

Therefore, if y_{ℓ_k} is lifted before $y_{\ell_{k-1}}$, the coefficient for $y_{\ell_{k-1}}$ is 0.

Condition 2. $y_{\ell_{k-1}}$ is lifted before y_{ℓ_k}. With Lemma 1, after lifting back $y_{\ell_{k-1}}$, we obtain a facet-defining inequality (6–23) for $X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_{k-1}\})\}$ and the coefficient of $y_{\ell_{k-1}}$ is $\beta_{\ell_{k-1}} = \min\{s + \sum_{j \in \mathcal{V}_\mathcal{R}} \beta_j y_j - \gamma : (s, y) \in X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_{k-1}\})\}$. Then, we lift $y_{\ell_{k-1}}$ based on (6–23) and obtain a facet-defining inequality. The coefficient of y_{ℓ_k} is

$$\alpha_{\ell_k} = \min\{s + \sum_{j \in \mathcal{V}_\mathcal{R}} \beta_j y_j + \beta_{\ell_{k-1}} y_{\ell_{k-1}} - (\gamma + \beta_{\ell_{k-1}}) : (s, y) \in X_{SDK}(\mathcal{V}_\mathcal{R} \cup \{\ell_{k-1}\} \cup \{\ell_k\})\}.$$

We have shown that all feasible solutions for $X_{SDK}(\mathcal{V}_{\mathcal{R}} \cup \{\ell_k\})$ are feasible for $X_{SDK}(\mathcal{V}_{\mathcal{R}} \cup \{\ell_{k-1}\} \cup \{\ell_k\})$. And $y_{\ell_{k-1}} = 0$ is feasible for $X_{SDK}(\mathcal{V}_{\mathcal{R}} \cup \{\ell_{k-1}\} \cup \{\ell_k\})$. Thus,

$$\begin{aligned}
\alpha_{\ell_k} =& \min\{s + \sum_{j \in \mathcal{V}_{\mathcal{R}}} \beta_j y_j + \beta_{\ell_{k-1}} y_{\ell_{k-1}} - (\gamma + \beta_{\ell_k}) : (s, y) \in X_{SDK}(\mathcal{V}_{\mathcal{R}} \cup \{\ell_{k-1}\} \cup \{\ell_k\})\} \\
=& \min\{s + \sum_{j \in \mathcal{V}_{\mathcal{R}}} \beta_j y_j - \gamma + 0\beta_{\ell_{k-1}} - \beta_{\ell_k} : (s, y) \in X_{SDK}(\mathcal{V}_{\mathcal{R}} \cup \{\ell_k\})\} \\
=& \beta_{\ell_k} - \beta_{\ell_{k-1}}.
\end{aligned}$$

Hence, if $y_{\ell_{k-1}}$ is lifted before y_{ℓ_k}, the coefficient for y_{ℓ_k} will be $\beta_{\ell_k} - \beta_{\ell_{k-1}}$.

We conclude these two conditions and obtain the lifting dominant property as follows.

Proposition 6.6. *Lifting Dominant Property: For any pair of nodes $(\ell_{k-1}, \ell_k) \in \mathcal{C}_{\mathcal{R}}(i)$ such that $b_{\ell_k} \geq b_{\ell_{k-1}}$ and $a(\ell_k) \in \mathcal{P}(a(\ell_{k-1}))$,*

1. *If y_{ℓ_k} is lifted before $y_{\ell_{k-1}}$, then the coefficient for y_{ℓ_k} is β_{ℓ_k} and the coefficient for $y_{\ell_{k-1}}$ is 0.*
2. *If $y_{\ell_{k-1}}$ is lifted before y_{ℓ_k}, then the coefficient for $y_{\ell_{k-1}}$ is $\beta_{\ell_{k-1}}$ and the coefficient for y_{ℓ_k} is $\beta_{\ell_k} - \beta_{\ell_{k-1}}$.*

Based on the lifting dominant property, we apply the lifting scheme to $X_{SDK}(\mathcal{V}_{\mathcal{R}})$ with y_{ℓ_i}, $\ell_i \in \mathcal{C}_{\mathcal{R}}(i)$ sequentially.

Theorem 6.9. *If $s + \sum_{j \in \mathcal{V}_{\mathcal{R}}} \beta_j y_j \geq \gamma$ is a facet-defining inequality for $X_{SDK}(\mathcal{V}_{\mathcal{R}})$, then the following inequality*

$$s + \sum_{j \in \mathcal{V}_{\mathcal{R}}} \beta_j y_j + \sum_{\ell_k \in \Lambda(i)} (\beta_{\ell_k} - \beta_{\ell_{k-1}}) y_{\ell_k} \geq \gamma + \beta_{\ell_K} \tag{6–24}$$

is facet-defining for $X_{SDK}(\mathcal{V}_{\mathcal{R}} \cup \mathcal{C}_{\mathcal{R}}(i))$, where $\Lambda(i) \subseteq \mathcal{C}_{\mathcal{R}}(i)$ in which $0 = \beta_{\ell_0} \leq \beta_{\ell_1} \leq \ldots \leq \beta_{\ell_{k-1}} \leq \beta_{\ell_k} \leq \cdots \leq \beta_{\ell_K}$, $\ell_K = argmax\{b_k : k \in \mathcal{C}_{\mathcal{R}}(i)\}$ and $a(\ell_k) \in \mathcal{P}(a(\ell_{k-1}))$.

Proof. We prove this conclusion by induction, following the lifting sequence $\ell_1, \ell_2, \cdots, \ell_K$.

Step 1. We lift y_{ℓ_1} from the facet-defining inequality $s + \sum_{j \in \mathcal{V}_R} \beta_j y_j \geq \gamma$. According to Lemma 1, we know that $s + \sum_{j \in \mathcal{V}_R} \beta_j y_j + \beta_{\ell_1} y_{\ell_1} \geq \gamma$ is a facet-defining inequality for $\mathcal{V}_R \cup \{\ell_1\}$ and $\beta_{\ell_1} = \min\{s + \sum_{j \in \mathcal{V}_R} \beta_j y_j - \gamma : (s, y) \in X_{SDK}(\mathcal{V}_R \cup \{\ell_1\})\}$.

Step 2. We lift y_{ℓ_2} after the lifting of y_{ℓ_1}. Following the sequence dominant property, y_{ℓ_1} is lifted before y_{ℓ_2}, where $b_{y_{\ell_1}} \geq b_{y_{\ell_2}}$ and $a(\ell_2) \in \mathcal{P}(a(\ell_1))$. Thus, the coefficient of y_{ℓ_2} is $\beta_{\ell_2} - \beta_{\ell_1}$ and the corresponding new lifted facet-defining inequality is

$$\begin{aligned} s + \sum_{j \in \mathcal{V}_R} \beta_j y_j + \beta_{\ell_1} y_{\ell_1} + (\beta_{\ell_2} - \beta_{\ell_1}) y_{\ell_2} &\geq \gamma + \beta_{\ell_1} + \beta_{\ell_2} - \beta_{\ell_1} \\ &\geq \gamma + \beta_{\ell_2}. \end{aligned} \tag{6–25}$$

With $\beta_{\ell_0} = 0$, we rewrite (6–25) as

$$s + \sum_{j \in \mathcal{V}_R} \beta_j y_j + (\beta_{\ell_1} - \beta_{\ell_0}) y_{\ell_1} + (\beta_{ell_2} - \beta_{\ell_1}) y_{\ell_2} \geq \gamma + \beta_{\ell_2}.$$

Step j. After lifting y_{ℓ_j}, we generate a facet-defining inequality as

$$s + \sum_{j \in \mathcal{V}_R} \beta_j y_j + \sum_{1 \leq k \leq j} (\beta_{\ell_k} - \beta_{\ell_{k-1}}) y_{\ell_k} \geq \gamma + \beta_{\ell_j}. \tag{6–26}$$

Step $j+1$. Now based on (6–26), we lift $y_{\ell_{j+1}}$ and the corresponding coefficient of $y_{\ell_{j+1}}$ is

$$\alpha_{j+1} = \min\{s + \sum_{j \in \mathcal{V}_R} \beta_j y_j + \sum_{1 \leq k \leq j} (\beta_{\ell_k} - \beta_{\ell_{k-1}}) y_{\ell_k} - (\gamma + \beta_{\ell_j}) : (s, y) \in X_{SDK}(\mathcal{V}_R \cup \{\ell_1\} \cup \cdots \cup \{\ell_{j+1}\})\}.$$

Now we prove that $\alpha_{\ell_{j+1}} = \beta_{\ell_{j+1}} - \beta_{\ell_j}$. Because $a(\ell_{j+1}) \in \mathcal{P}(a(\ell_j))$, by induction, we have $a(\ell_{j+1}) \in \mathcal{P}(a(\ell_i))$, where $1 \leq i \leq j$. Thus if $(\widetilde{s}, \widetilde{y}) \in X_{SDK}(\mathcal{V}_R \cup \{\ell_{j+1}\})$, then $(\widetilde{s}, \widetilde{y}) \in X_{SDK}(\mathcal{V}_R \cup \{\ell_{j+1}\} \cup \{\ell_1\} \cup \cdots \cup \{\ell_j\})$. Hence,

$$\begin{aligned} \alpha_{j+1} &= \min\{s + \sum_{j \in \mathcal{V}_R} \beta_j y_j + \sum_{1 \leq k \leq j} (\beta_{\ell_k} - \beta_{\ell_{k-1}}) y_{\ell_k} - (\gamma + \beta_{\ell_j}) : (s, y) \in X_{SDK}(\mathcal{V}_R \cup \{\ell_1\} \cup \cdots \cup \{\ell_{j+1}\})\} \\ &= \min\{s + \sum_{j \in \mathcal{V}_R} \beta_j y_j - \gamma + \sum_{1 \leq k \leq j} (\beta_{\ell_k} - \beta_{\ell_{k-1}}) y_{\ell_k} - \beta_{\ell_j} : (s, y) \in X_{SDK}(\mathcal{V}_R \cup \{\ell_1\} \cup \cdots \cup \{\ell_{j+1}\})\} \\ &= \beta_{\ell_{j+1}} - \beta_{\ell_j}. \end{aligned}$$

Thus, the above conclusion holds by induction. □

6.5 Cutting Plane Generation under Parallel Environment

Our parallel implementation involves assigning slave processors to generate cuts for the capacitated stochastic lot-sizing problem (CSLS). The purpose of our parallel implementation is to test the effectiveness and efficiency of our cuts based on pairing, lifting scheme. In order to evaluate the the performance of our cuts, we implement two parallel schemes for CSLS: 1). the cut-and-branch algorithm, as well described further in section 6.5.3, and 2). the LP-based cutting plane algorithm with heuristic, as well described further in section 6.5.4. The major issues of parallel implementation is that of load balancing among processors and management of generated cuts. The desirable condition is that each processor handles approximately an equal number of LPs and the cutting management can control the cutting pool size, recognize the strong cuts, and purge the ineffective cuts.

Our parallel implementation was developed on a Pentium4 Xeon64 quad core Linux cluster and distributed-memory multiprocessors. We apply the open source software SYMPHONY in COIN-OR as our mixed-integer programming solver. We let κ and ω represent the total number of processors and the slave processors working on the cut generations.

In order to generate our cuts, we use multiprocessors to generate cuts according to the stochastic scenario tree structure. This section is organized as follows. In Section 6.5.1, we discuss the static partition for the stochastic scenario tree structure. In Section 6.5.2, we discuss the cut management of our two algorithms in a parallel distributed environment. In Sections 6.5.3 and 6.5.4, we provide the detailed parallel algorithm to obtain the optimality and integrality gaps for the stochastic integer programming problems.

6.5.1 Partitions of Stochastic Scenario Tree and Local Initialization

We develop two algorithms for a stochastic integer programming problem with stochastic scenario tree model. In order to handle the load balance issue, we discuss static partitioning for the stochastic scenario tree structure. After the partition, each processor handles a stochastic scenario subtree to generate cuts.

Static Partitioning: We partition the stochastic scenario tree into several subtrees. The partition is based on the total number of leaf nodes and each partition is recognized by the index of leaf nodes, i.e., the root node of the scenario tree is the root node of each subtree. Its leaf nodes are indexed from $(\mathcal{L}/\omega) * (i-1)+1$ to$(\mathcal{L}/\omega) * i$. Thus, the total partition number is $|\mathcal{L}/\omega|$ and each processor handles a $|\mathcal{V}| \times \omega/|\mathcal{L}|$ subtree. Note that the local initialization on the processor only needs to load a physical copy of the corresponding subtree structure.

6.5.2 The Parallel Cuts Management Decision Control

6.5.2.1 Cut Generation

For an individual processor, the cut generation involves generating cuts based on each subtree structure. In Section 6.3 and 6.4, there are three cut generation schemes, the path lifting scheme, the pairing scheme, the tree lifting scheme. These three scheme cuts are generated on each processor.

For the static partitioning, first, each processor obtains the information of the subtree from the master machine. Second, the corresponding basic path inequalities are generated based on the subtree structure. Third, the path lifting inequalities are generated. Fourth, the pairing scheme is applied to the subtree structure to generate valid inequalities. Finally, the lifting scheme is applied to the pairing inequalities.

Due to each processor should handle these three types of inequality, we assign time limits for each scheme for cut generation. If a scheme of cut generation obtained long time intervals, the processor spends longer time on generating cuts based on the scheme.

6.5.2.2 Cutting Pool Management

All generated cuts are stored locally. Before, each processor sends all generated cuts to the master processor, local cutting pool management function is called. The local cutting pool purges the duplicated cuts from the cutting pool.

The master machine handles the master cutting pool which obtained all cuts from the local cutting pool and purge the duplicated cuts before all cuts are applied to the process in the master machine.

6.5.3 The Parallel Cut-and-Branch Algorithm

First, we discuss a straightforward algorithm for CSLS, the parallel cut-and-branch algorithm. We use multiprocessors to generate as many cuts as possible for CSLS within a certain time limit. Then SYMPHONY uses the branch-and-bound algorithm to solve a big MIP problem which combines all generated cuts with the original CSLS problem. The detailed algorithm is described as follows:

The parallel cut-and-branch algorithm

1. The cut generation: generate cuts based on the subtree of the stochastic scenario tree according to the leaf nodes.

2. The cuts management: purge the duplicated cuts.

3. The branch-and-bound algorithm for CSLS with all generated cuts in the cutting pool.

Step 1 handles the partitions for a stochastic scenario tree. The cut generation is from the shortest branch to the longest branch starting from the root nodes. First, we generate the basic inequalities. Second, we generate the lifting inequalities based on the basic inequalities. Third, we generate the pairing inequalities based on the generated basic and lifted inequalities. In Step 2, we apply the basic cut management for the generated cuts. The purpose of cut management is to purge the duplicated cuts and store all efficient cuts. In Step 3, we combine all generated cuts with the original CSLS problem as a big MIP problem and using the branch-and-bound scheme in

SYMPHONY to solve within the time limits. Finally, we obtain the optimality gap, as the evaluation standard, from the parallel cut-and-branch algorithm.

6.5.4 The LP Based Cutting Plane Algorithm

The parallel cut-and-branch algorithm generates cuts at the beginning and sends all cuts to the branch-and-bound structure. Now we discuss a LP based cutting plane algorithm with an embedded heuristics to obtain an integer feasible solution.

The LP based cutting plane algorithm

Step 1. Set $k = 1$. Let the upper bound of the problem be $U_{IP} = +\infty$ and the lower bound be $L_{IP} = -\infty$. Let the initial integer feasible solution be $x_{IP} = 1$. Let S_R represent the feasible region of LP relaxation and $S_R^1 = S_R$.

Step 2. Solve the relation problem of S_R^t and obtain the solution x^t and objective z_{LP}

Termination tests: If x^t is an integer soltion, stop. x^t is the optimal solution of $CSLS$.

Step 3. Heuristics: call Heuristics for obtaining the integer feasible solution to obtain the integer feasible solution, x_H. If $z_H \leq U_{IP}$, $U_{IP} = z_H$ and $x_{IP} = x_H$. Otherwise, $N = N + 1$, where N represents the number of heuristics calls.

Termination test: If $N \geq K$, stop.

Step 4. Update $L_{IP} = z_{LP}$ and S^{k+1} from S^k with more generated cuts.

Heuristics for obtaining the integer feasible solution

Step 1. Let $x_{max}^t = \{x_j^t, x_j^t \geq \varepsilon_1\}$.

Step 2. For $x_j^t \geq \varepsilon_1$, if $x_j^t \geq x_{max}^t - \varepsilon_2$, $x_j^t = 1$.

Before the implementation of this heuristics, the synchronization (share) of cuts is called among processors and the corresponding LP problem (S_R^t) is processed. After the implementation of this Heuristics, the modified LP with fixed integer variables is resolved. Then this heuristics is called again until an integer feasible solution is generated.

6.6 Computational Results

In this section, we report the computational results to demonstrate the computational efficiency of the three types of cuts for stochastic dynamic knapsack sets.

6.6.1 Instance Generation

We consider the stochastic capacitated lot-sizing problem as an example for the stochastic dynamic knapsack sets. The stochastic capacitated lot-sizing problem based on the stochastic scenario tree setting can be formulated as follows:

$$\begin{aligned}
\min \quad & \sum_{i \in \mathcal{V}} p_i(\alpha_i x_i + h_i s_i + f_i y_i) \\
s.t. \quad & x_i + s_{i^-} = d_i + s_i && i \in \mathcal{V} \\
& x_i \leq c_i y_i && i \in \mathcal{V} \\
& x_i \geq 0, y_i \in \{0, 1\} && i \in \mathcal{V},
\end{aligned}$$

where decision variables x_i, s_i are the production and inventory amounts at node i. Binary decision variable y_i is the setup decision at node i. Parameters d_i and c_i are the demand and capacity of node i. Parameter p_i is the probability associated with the realization of each scenario.

In the following, we consider two stochastic scenario trees. The first is a binary tree with sixteen periods (P2-16). The second one is a tree with three branches at each non-leaf nodes and thirteen periods (P3-13). For each tree, we set the ratio $f/\alpha = 10, 20, 30$ and 40. We generate two instances with ratio 10. There are ten combinations in total. We also set demands d_i, unit production cost α_i, and unit inventory cost h_i uniformly distributed in [50, 50], [50, 100], [5,10]. For each setting, we test five instances and report the average value.

6.6.2 The Cut-and-Branch Algorithm

We show the performance of the cut-and-branch algorithms for the stochastic capacitated lot-sizing problem by the following figures. We evaluate the performance

Table 6-1. Parameter setting

		unit production cost p_i	setup cost f_i
1	ratio 10	[50, 100]	[500, 1000]
2	ratio 20	[50, 100]	[1000, 2000]
3	ratio 30	[40, 60]	[1000, 2000]
4	ratio 40	[40, 60]	[500, 1000]
5	ratio 10	[50, 50]	[500, 500]

of the cut-and-branch algorithm by the gap percentage $O_{Gap} = \frac{UB-LB}{UB} \times 100\%$, where "UB" is the objective value corresponding to the best integer solution obtained in the given time limit. "LB" is the lower bound obtained from linear programming relaxation. Figure 6-2 and Figure 6-4 show the optimality gaps for (P2-16) and (P3-13) with the high setup costs with 5 ratios. For (P2-16), with more generated cuts, O_{Gap} is decreased from 6.18% to 2.10% in average. For (P3-13), O_{Gap} is decreased from 7.99% to 3.45% in average. Figure 6-3 and Figure 6-5 show the corresponding cuts numbers generated for both cases.

6.6.3 The LP Based Heuristics

In this section, we show the performance of the LP based heuristics for the cut generation. We use the integrality gap I_{Gap} to evaluate the performance of cut generation, where $I_{Gap} = \frac{z_{LP}^{H} - z_{IP}}{z_{LP} - z_{IP}} \times 100\%$. In Table 6-2, we provide the average of all five parameter settings for (P2-16) and (P3-13). We can see that with the cuts generation and heuristics, we provide integer feasible solutions and improve the integrality gaps.

Table 6-2. Heuristics for low setup cost case

	LP	MIP	HEUR	IGAP %
2-16	1.182×10^6	1.788×10^6	2.062×10^6	45.27%
3-13	9.582×10^6	1.335×10^7	1.711×10^7	52.90%

Figure 6-2. Gaps for P2-16

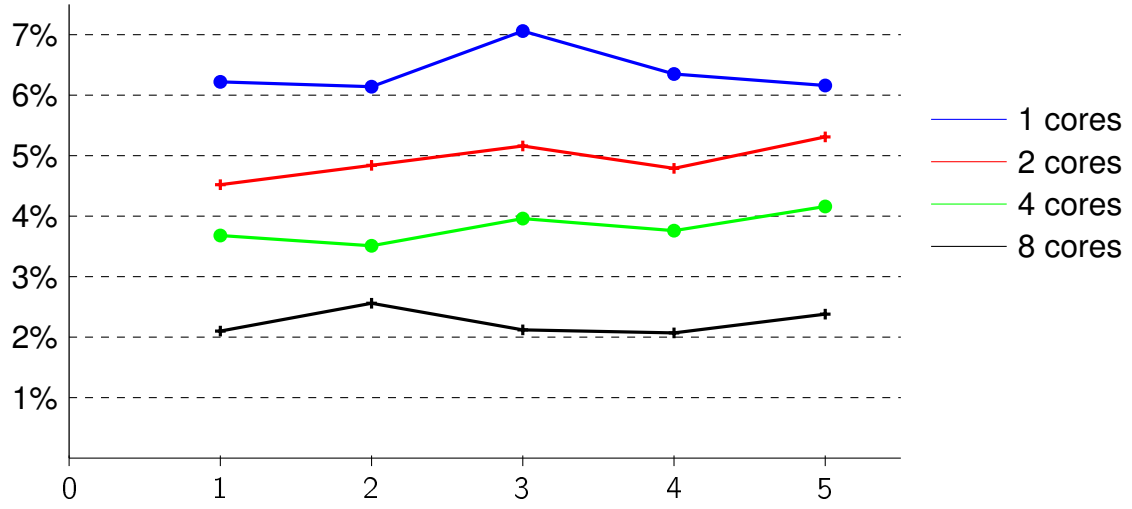


Figure 6-3. Cuts number for P2-16

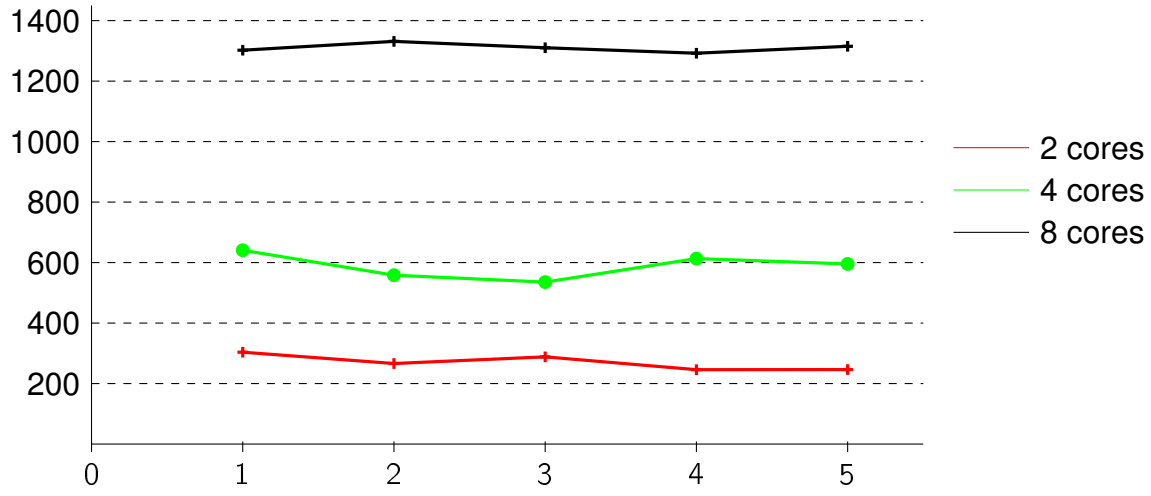


Figure 6-4. Gaps for P3-13

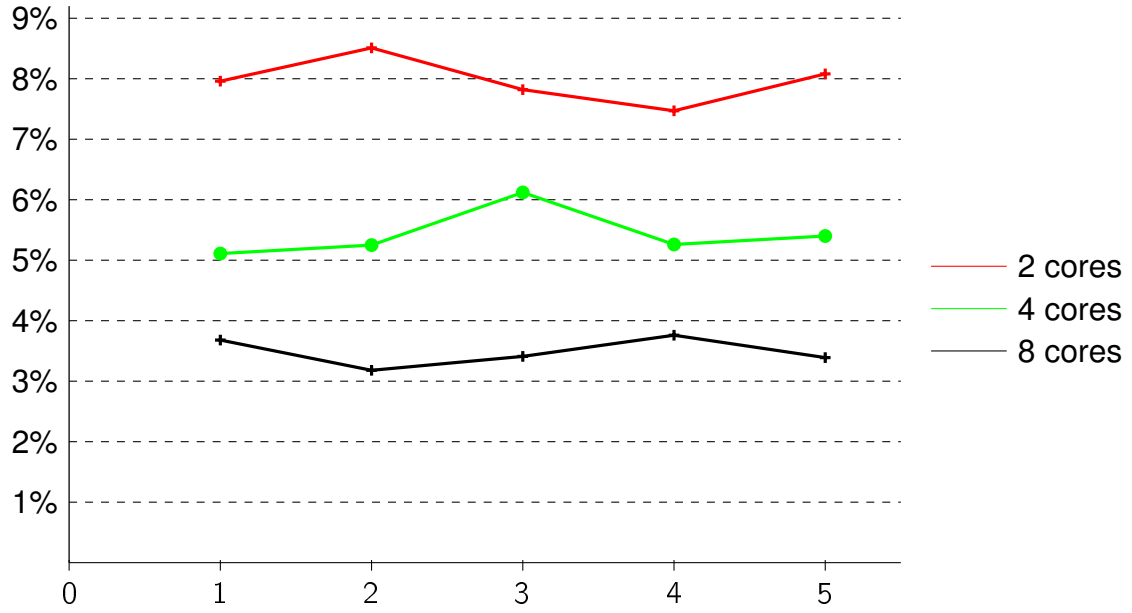
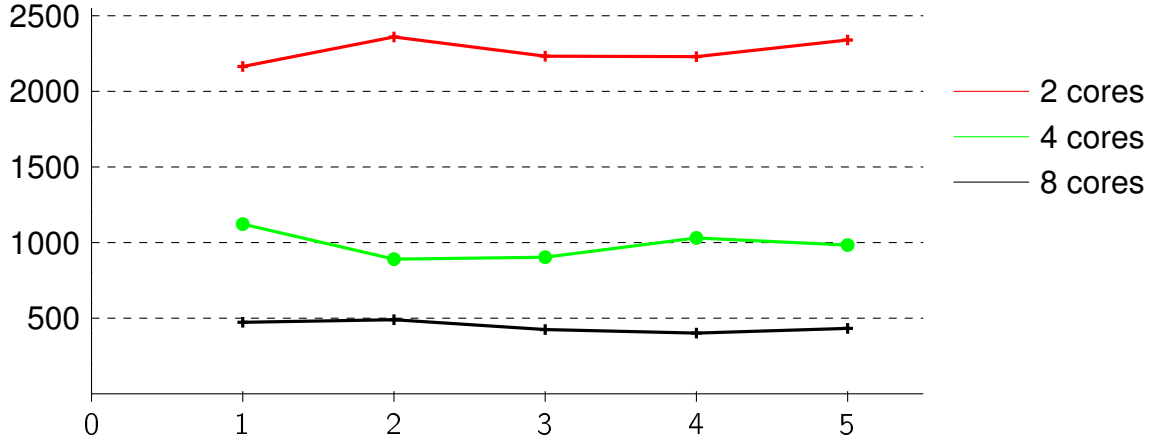


Figure 6-5. Cuts number for P3-13



CHAPTER 7
CONCLUSION

In this dissertation, we discussed the multi-stage discrete optimization problem under data uncertainty. We studied the multi-stage robust lot-sizing with disruptions, the polyhedron of two-stage stochastic lot-sizing problems, and the generation of valid inequalities for the stochastic dynamic knapsack set. In Chapter 2, we studied the lot-sizing problem with a potential disruption as recourse to handle the uncertainty. Our objective is to achieve the minimum objective value that considers the worst case scenario for the disruption. We developed a customized branch-and-bound algorithm to solve the lot-sizing problem with a disruption, which is a two-stage robust optimization problem. A Benders' decomposition based optimality test is generated for the branch-and-bound algorithm. This study provides more robust production planning to address a disruption, as compared to the deterministic lot-sizing problem and previous studies on recovery production with the information of disruption.

In Chapter 3, we provided a general model for the multi-stage robust lot-sizing problem with uncertain disruptions. We adopted outsourcing and backlogging as the reparation methods to satisfy unfilled demands due to disruptions. For the outsourcing case, based on a proper assumption, a two-stage robust model and the corresponding primal-dual algorithm are generated. For the backlogging case, we first considered a two-stage robust optimization problem. We reformulated this problem as a mixed-integer program and investigated the corresponding polyhedral structure. We analyzed the trivial facet-defining inequalities, and generated three families of facet-defining inequalities for the polyhedron of the robust lot-sizing problem with disruption and backlogging. Applying these facet-defining inequalities as cuts, the computational results demonstrate that these inequalities accelerate the computational speed as compared with default CPLEX.

In our multi-stage robust optimization setting, we considered the multi-stage robust lot-sizing problem with multiple disruptions. We studied the multi-stage robust lot-sizing problem without and with setup cost, respectively. We proposed a reformulation scheme to transfer the multi-stage robust optimization problem to be a mixed-integer linear program, and provided the theoretical proof to show that it is one-to-one correspondence between the reformulation and the original multi-stage robust optimization model. Thus, the multi-stage robust lot-sizing problem with multiple disruptions is computationally tractable. For the non-setup cost case, we generated a reformulated minimization linear program with a pre-processing algorithm. We proved that the reformulation scheme also works for the multi-stage robust mixed-integer program. As compared to the two-stage robust optimization model, the multi-stage model is much more challenging to solve. The reformulation scheme proposes a solution approach for the multi-stage robust mixed-integer program.

Chapters 4 and 5 contribute to the literature on deriving integral polyhedral descriptions and extended formulations for multi-period stochastic uncapacitated lot-sizing problems. To the best of our knowledge, there is no previous research on developing an extended formulation which provides integral solutions for the multi-period stochastic uncapacitated lot-sizing problem. In Chapter 4, we introduce a deterministic equivalent formulation for a two-stage stochastic uncapacitated lot-sizing problem with Wagner-Whitin costs and deterministic demands. We examined the optimal solution property and used it to generate an extended formulation in the higher dimensional space. Then, we proved that the constraint matrix for the extended formulation is totally unimodular. Finally, we projected the extended formulation back to the original space so that we can find valid inequalities which describe the integral polyhedron of the problem in the original space with $\mathcal{O}(|\mathcal{V}|)$ variables and $\mathcal{O}(TW|\mathcal{V}|)$ constraints, where $|\mathcal{V}|$ is the cardinality of $\mathcal{V}$.

In Chapter 5, we studied the two-stage stochastic lot-sizing problem with backlogging and Wagner-Whitin costs. We investigated the relationship among the setup, inventory, and backlogging decisions and obtained an extended formulation. We proved that the extended formulation provides integral solutions for this problem. Further, we generated another extended formulation in lower dimension space by projection. With the polyhedral descriptions and the extended formulations for two-stage stochastic lot-sizing problems, the practical tactical production decision problems can be solved by linear programs with $\mathcal{O}(|\mathcal{V}|)$ variables and $\mathcal{O}(TW|\mathcal{V}|)$ constraints.

In Chapter 6, we studied the extension of the deterministic dynamic knapsack sets, the stochastic dynamic knapsack set, and investigated the polytope of the stochastic dynamic knapsack set with a scenario tree model. We studied the pairing scheme, mixing scheme, lifting scheme for the stochastic dynamic knapsack set to generate valid inequalities, which are facet-defining under certain conditions. The stochastic scenario tree model involves many scenarios and causes computational difficulties. Therefore, we applied parallel computing to solve the stochastic dynamic knapsack set. We developed parallel computing algorithms to solve the stochastic capacitated lot-sizing problem as an example of the stochastic dynamic knapsack set.

For the multi-stage discrete optimization under uncertainty problem, the multi-stage robust optimization and stochastic programming are two main approaches. In the future research, there are other interesting settings and models to study. For the multi-stage robust optimization model, in Chapter 3, we studied the polyhedral structure of a two-stage robust mixed-integer program. There is potential to extend current results to the multi-stage robust mixed-integer model, and generate the corresponding facet-defining inequalities to describe the polyhedron of the multi-stage robust mixed-integer program. We can also consider a more general setting for the multi-stage robust mixed-integer program, in which the uncertain parameters are in given intervals.

For the more general setting, whether the multi-stage robust optimization is still tractable and how to generalize the reformulation scheme are interesting research topics.

For the multi-stage stochastic programming, a part of our results for the two-stage stochastic lot-sizing can be applied to a more general multi-stage stochastic programming setting, which can address further uncertainties. Under the multi-stage setting, it can be observed that the optimality condition still holds, based on the Wagner-Whitin costs setting for cases without and with backlogging. Accordingly, we can obtain similarly constraints for the reformulation. Whether the reformulation can provide an extended formulation that provides integral solutions for the multi-stage stochastic uncapacitated lot-sizing problem is also of interest for future study.

REFERENCES

Aggarwal, A., J. Park. 1993. Improved algorithms for economic lot-size problem. *Operations Research* **41** 549–571.

Ahmed, S., A. J. King, G. Parija. 2003. A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization* **26** 3–24.

Ahmed, S., A. Shapiro. 2002. The sample average approximation method for stochastic programs with integer recourse. Tech. rep., The Georgia Institute of Technology.

Atamtürk, A., M. Zhang. 2007. Two-stage robust network flow and design under demand uncertainty. *Operations Research* **55** 662–673.

Atamtürk, Alper. 2004. Sequence independent lifting for mixed-integer programming. *Operations Research* **52** 487–490.

Baker, K. R., P. Dixon, M. J. Magazine, E. A. Silver. 1978. An algorithm for the dynamic lot-size problem with time-varying production capacity constraints. *Management Science* **24** 1710–1720.

Barany, I., T. van Roy, L. A. Wolsey. 1984a. Strong formulations for multi-item capacitated lot sizing. *Management Science* **30** 1255–1262.

Barany, I., T. J. van Roy, L. A. Wolsey. 1984b. Uncapacitated lot sizing: the convex hull of solutions. *Mathematical Programming Study* **22** 32–43.

Beale, E. M. L. 1955. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society* **17** 173–184.

Beale, E. M. L., G. B. Dantzig, R. D. Watson. 1986. A first order approach to a class of multi-time-period stochastic programming problem. *Mathematical Programming Study* **27** 103–117.

Ben-Tal, A., A. Goryashko, E. Guslitzera, A. Nemirovski. 2004. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming* **99** 351–376.

Ben-Tal, A., A. Nemirovski. 1998. Robust convex optimization. *Mathematics of Operations Research* **23** 769–805.

Ben-Tal, A., A. Nemirovski. 2000. Robust solution of linear programming problems contaminated with uncertain data. *Mathematical Programming* **88** 411–424.

Benders, J. F. 1962. Partitioning procedures for solving mixed-variable programming problems. *Numerische Mathemstik* **4** 238–252.

Bertsimas, D., D. Pachamanova, M. Sim. 2004. Robust linear optimization under general norms. *Operations Research Letters* **32** 510–516.

Bertsimas, D., M. Sim. 2003. Robust discrete optimization and network flows. *Mathematical Programming* **98** 49–71.

Bertsimas, D., M. Sim. 2004. The price of robustness. *Operations Research* **52** 35–53.

Bertsimas, D., D. Simchi-Levi. 1996. A new generation of vehicle routing research: Robust algorithm, addressing uncertainty. *Operations Research* **44** 286–304.

Bertsimas, D., A. Thiele. 2006. A robust optimization approach to inventory theory. *Operation Research* **54** 150–168.

Bienstock, D., N. Özbay. 2008. Computing robust basestock levels. *Discrete optimization* **5** 389–414.

Birge, J. R. 1985. Decomposition and partitioning method for multi-stage stochastic linear programs. *Operations Research* **33** 989–1007.

Birge, J. R. 1988. The relationship between the L-shaped method and dual basis factorization for stochastic linear programming. Y. Ermoliev, R. Wets, eds., *Numerical Techniques for Stochastic Optimization*. Springer Verlag, Berlin.

Birge, J. R., F. Louveaux. 1997. *Introduction to Stochastic Programming*. Springer Verlag, Berlin.

Carlson, R., J. Jucker, D. Kropp. 1979. Less nervous MRP systems: A dynamic economic lot-sizing approach. *Management Science* **25** 754–761.

Dantzig, G. B., P. Wolfe. 1960. Decomposition principle for linear programs. *Operations Research* **8** 101–111.

Danzig, G. B. 1955. Linear programming under uncertainty. *Management Science* **1** 197–206.

Danzig, G. B., A. Mandasky. 1961. On the solution of two-stage linear programs under uncertainty. *4th Berkeley Symposium on Mathematics Statistics and Probability*. University of California Press, Berkeley.

Federgrun, A., M. Tsur. 1991. A simple forward algorithm to solve general dynamic lot-size models with n period in $\mathcal{O}(nlogn)$ or $\mathcal{O}(n)$ time. *Management Science* **37** 909–925.

Fisher, M., K. Ramdas, Y. Zheng. 2001. Ending inventory valuation in multiperiod production scheduling. *Management Science* **47** 679–692.

Grinold, R. C. 1976. A new approach to multi-stage stochastic linear programs. *Mathematical Programming Study* **6** 19–29.

Gu, Z., G. L. Nemhauser, M. Savelsbergh. 2000. Sequence independent lifting in mixed integer programming. *Journal of Combinatorial Optimization* **4** 1573–2886.

Guan, Y., S. Ahmed, A. J. Miller, G. L. Nemhauser. 2006a. On formulations of the stochastic uncapacitated lot-sizing problem. *Operations Research Letters* **34** 241–250.

Guan, Y., S. Ahmed, G. L. Nemhauser. 2007. Sequential pairing of mixed integer inequalities. *Discrete Optimization* **4** 21–39.

Guan, Y., S. Ahmed, G. L. Nemhauser. 2009. Cutting planes for multistage stochastic integer programs. *Operations Research* **57** 287–298.

Guan, Y., S. Ahmed, G. L. Nemhauser, A. J. Miller. 2006b. A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem. *Mathematical Programming* **105** 55–84.

Guan, Y., A. J. Miller. 2008. Polynomial time algorithms for stochastic uncapacitated lot-sizing problems. *Operations Research* **56** 1172–1183.

Günlük, O., Y. Pochet. 2001. Mixing mixed-integer inequalities. *Mathematical Programming* **90** 429–457.

Hartman, J. C., I. E. Büyüktahtakin, J. C. Smith. 2010. Dynamic programming based inequalities for the capacitated lot-sizing problem. *IIE Transactions* .

Huang, K., S. Küçükyavuz. 2008. On stochastic lot-sizing problems with random lead times. *Operations Research Letters* **36** 303–308.

Kall, P. 1976. *Stochastic Linear Programming*. Springer Verlag, Berlin.

Kazan, O., R. Nagi, C. M. Rump. 2000. New lot-sizing formualtions for less nervous production schedules. *Computers and Operations Research* **27** 1325–1345.

Kouvelis, P., G. Yu. 1997. *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers, Norwell.

Krarup, J., O. Bilde. 1977. Plant location, set covering and economic lot sizes: an $\mathcal{O}(mn)$ algorithm for structured problems. L. Collatz, ed., *Optimierung bei Graphentheoretischen und Ganzzahligen Probleme*. Birkhauser Verlag, Basel, 155–180.

Küçükyavuz, S., Y. Pochet. 2007. Uncapacitated lot sizing with backlogging: the convex hull. *Mathematical Programming* **118** 151–175.

Lee, Y. M., E. J. Chen. 2005. Case studies: Supply chain optimization models in a chemical company. J. Geunes, E. Akcali, P. M. Pardalos, H. E. Romeijn, Z. J. Shen, eds., *Applications of Supply Chain Management and E-Commerce Research*. Springer, 453–477.

Loparic, M., H. Marchand, L. A. Wolsey. 2003. Dynamic knapsack sets and capacitated lot-sizing. *Mathematical Programming* **95** 53–69.

Louveaux, F. 1980. A solution method for multistage stochastic programs with recourse with application to an energy investment problem. *Operations Research* **28** 889–902.

Louveaux, F., R. Schultz. 2003. Stochastic integer programming. A. Ruszczyński, A. Shapiro, eds., *Handbooks in Operations Research and Management Science: Stochastic Programming*. Elsevier, 213–266.

Martin, R. 1987. Generating alternative mixed-integer programming models using variable redefinition. *Operations Research* **35** 820–831.

Miller, A., L. A. Wolsey. 2003. Tight Mip formulation for multi-item discrete lot-sizing problems. *Operations Research* **51** 557–565.

Nemhauser, G. L., L. A. Wolsey. 1999. *Integer and Combinatorial Optimization*. Wiley.

Pereira, M. V. F., L. M. V. G. Pinto. 1985. Stochastic optimization of a multireservoir hydroelectric system: a decomposition approach. *Water Resources Research* **21** 779–792.

Pochet, Y., L. A. Wolsey. 1988. Lot-size models with backlogging: Strong reformulations and cutting planes. *Mathematical Programming* **40** 317–335.

Pochet, Y., L. A. Wolsey. 1994. Polyhedra for lot-sizing with Wagner-Whitin costs. *Mathematical Programming* **67** 297–323.

Radin, R., U. Choe. 1979. Tighter relaxations of fixed charge network flow problems. Tech. rep., The Georgia Institute of Technology, Atlanta, Georgia.

Ruszczyński, A., A. Shapiro, eds. 2003. *Handbooks in Operations Research and Management Science: Stochastic Programming*, vol. 10. Elesvier.

Shapiro, A. 2000. *Stochastic Programming by Monte Carlo Simulation Methods*. Stochastic Programming E-Print Series.

Shapiro, A., T. Homem de Mello. 2001. On rate of convergence of monte carlo approximations of stochastic programs. *SIAM Journal on Optimization* **11** 70–86.

Soyster, A. L. 1973. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research* **2** 1154–1157.

Strazicky, B. 1962. Some results concerning an algorithm for the discrete recourse problem. *Stochastic Programming* **4** 238–252.

Summa, M. Di, L. A. Wolsey. 2006. Lot-sizing on a tree. *Operations Research Letters* **105** 55–84.

van Hoesel, C., A. Wagelmans, L. A. Wolsey. 1994. Polyhedral characterization of the economic lot-sizing problem with start-up costs. *SIAM Journal on Discrete Mathematics* **7** 141–151.

Van Slyke, R. M., R. J. B. Wets. 1969. Programming under uncertainty and stochastic optimal control. *SIAM Journal on Applied Mathematics* **17** 638–663.

Wagelmans, A., C. van Hoesel, A. Kolen. 1992. Economic lot sizing: An $\mathcal{O}(n \log n)$ algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research* **40** 145–156.

Wagner, H. M., T. M. Whitin. 1958. Dynamic version of the economic lot size model. *Management Science* **5** 89–96.

Wets, R. J. B. 1974. Stochastic programs with fixed recourse: The equivalent deterministic program. *SIAM Review* **16** 309–339.

Wets, R. J. B. 1983. Solving stochastic programs with simple recourse. *Stochastics* **10** 219–242.

Wolsey, L. A. 1998. *Integer Programming*. Wiley.

Wolsy, L. A. 1976. Facets and strong valid inequalities for integer programs. *Operations Research* **24** 367–372.

Wolsy, L. A. 1977. Valid inequalities and superadditivity for 0/1 integer programs. *Mathematics of Operations Research* **2** 66–77.

Yang, J., X. Qi, G . Yu. 2005. Disruption management in production planning. *Naval Research Logistics* **52** 420–442.

Zemel, E. 1978. Lifting the facets of o-1 polytopes. *Mathematical Programming* **5**.

BIOGRAPHICAL SKETCH

Zhili Zhou was born in 1981, Nanjing, Jiangsu, China to parents Guifeng Wang and Yaozong Zhou. She was raised in Nanjing and attended Nanjing No.3 Middle school for her middle school and high school education. She went on to attend college at Nanjing University in Nanjing, China, in 1981, majoring in Information and Computational Mathematics. She received her bachelor's degree in science in June 2003. Zhili then elected to continue her education in Department of Mathematics, Nanjing University, majoring in Operations Research and completed her master degree in science in June 2006.

Zhili enrolled in the Ph.D. programming in the School of Industrial Engineering, University of Oklahoma, in 2006 and transferred to Department of Industrial and Systems Engineering, University of Florida, in 2009 with her advisor, Dr. Yongpei Guan to obtain a Doctor of Philosophy degree.